R. F. Matthaei

BASIC

für blutige Laien

ISBN 3–921682–90–8

Es kann keine Gewähr dafür übernommen werden, daß die in diesem Buche verwendeten Angaben, Schaltungen, Warenbezeichnungen und Warenzeichen, sowie Programmlistings frei von Schutzrechten Dritter sind. Alle Angaben werden nur für Amateurzwecke mitgeteilt. Alle Daten und Vergleichsangaben sind als unverbindliche Hinweise zu verstehen. Sie geben auch keinen Aufschluß über eventuelle Verfügbarkeit oder Liefermöglichkeit. In jedem Falle sind die Unterlagen der Hersteller zur Information heranzuziehen.

Nachdruck und öffentliche Wiedergabe, besonders die Übersetzung in andere Sprachen verboten. Programmlistings dürfen weiterhin nicht in irgendeiner Form vervielfältigt oder verbreitet werden. Alle Programmlistings sind Copyright der Fa. Ing. W. Hofacker GmbH. Verboten ist weiterhin die öffentliche Vorführung und Benutzung dieser Programme in Seminaren und Ausstellungen. Irrtum, sowie alle Rechte vorbehalten.

COPYRIGHT BY ING. W. HOFACKER © 1983
Tegernseerstr. 18, 8150 Holzkirchen

1. Auflage 1983
Gedruckt in der Bundesrepublik Deutschland – Printed in West-Germany – Imprime'en RFA.

Inhaltsverzeichnis

Praktische Hinweise 1

1. LEKTION 3
- Der Computer 3
- Die Betriebsarten (Modes) 4
- Die Tastatur im Befehlsmodus 5
- Die Befehlsinterpretation im Befehlsmodus 6
- Übung zu FOR-TO-NEXT 8
- Übung zur Laufzeitermittlung 10

2. LEKTION 11
- Variablennamen und Felder 11
- Felder 13
- Eindimensionale Felder 13
- Beginn der 1. Fallstudie: Durchschnitt errechnen 14
- Der Befehl END 16
- Der Befehl RUN 16

3. LEKTION 17
- Plausibilieren von Tastatureingaben 17
- GOTO Zeilennummer 20
- Fortsetzung der 1. Fallstudie 21

4. LEKTION 25
- Der Befehl IF 25
- THEN ohne ELSE 25
- THEN mit ELSE 26
- Korrekturmöglichkeiten beim Fallstudienprogramm 27
- GOSUB und RETURN 28
- Musterlösung zur 1. Fallstudie 30

5. LEKTION 33
- Zwei- und mehrdimensionale Felder 33
- Verschachtelte Schleifen 33
- DATA, READ und RESTORE 34
- 2. Fallstudie, Aufgabenstellung 36

6. LEKTION **41**
Editieren (Erfassen) von Programmen 41
Fortsetzung der 2. Fallstudie 42

7. LEKTION **45**
Strings 45
Der Befehl VAL 47
Generelles zur Laufzeit 48
Fortsetzung der 2. Fallstudie 49
Plausibilieren von Daten im Programmtext 49

8. LEKTION **53**
Bubblesort 55
Sortieren durch direktes Einfügen 58
Sortieren durch direktes Auswählen 62

9. LEKTION **67**
Ende der 2. Fallstudie 67
INKEY$ 70
ASC(string) 70
CHR$(num) 71
Die Tabulation über Platzsparcodes 72
Strukturierung 76
A. Die FOR-Schleife 77
B. REPEAT ... UNTIL 77
C. WHILE ... DO 78
D. IF ... THEN ... ELSE 79
E. CASE 79
Direktspeicherzugriffsbefehle PEEK und POKE 81
Der Video-RAM und PEEK 82
Drucker-DCB 82
ON ... GOTO und ON ... GOSUB 84
Der Befehl INPUT 85
Rechen(un)genauigkeiten 86
PRINT USING 87
Vorbereitung auf die 3. Fallstudie 87
RANDOM 88
RND (wert) 88
Aufgabenstellung zur 3. Fallstudie 89

10. LEKTION **91**
Ende der 3. Fallstudie 91

11. LEKTION **99**
Betrieb mit Disketten 99
BASIC/CMD 101
Dateinamen 105
Datensicherung 105
Auftreten von Lesefehlern 106
ANLAGE 1 ..... **109**
Beispiel für DIRCHECK/CMD 109
ANLAGE 2 ..... **110**
Beispiel für ein Kaltstart-Hilfsprogramm 110
ANLAGE 3 ..... **112**
Beispiel für ein Adressenverwaltungsprogramm 112

Zu diesem Buch

Zu diesem Buch

Dieses Lehr- und Übungsbuch ist für das Selbststudium gedacht und ist abgestimmt auf 11 Lektionen zu je etwa fünf bis acht Stunden Bearbeitungszeit. Es kann das Systemhandbuch nicht ersetzen, sondern setzt im Gegenteil das Vorhandensein des Systemhandbuches und eines Tischcomputers (vorzugsweise eines TRS-80 oder Video Genies) voraus.

Im Gegensatz zu herkömmlichen BASIC-Lehr- und Übungsbüchern enthält dieses Buch nur wenig Programmbeispiele, spricht diese aber sehr detailliert und unter Nutzung der interaktiven Möglichkeiten der Tischcomputer durch.

Zu diesem Verfahren ist ein Zitat von N. Wirth aus dem Vorwort seines Werkes "Algorithmen und Datenstrukturen" (1975) als Begründung angebracht:

"Programmieren ist eine konstruktive Tätigkeit. Wie kann eine aufbauende, schöpferische Fähigkeit gelehrt werden ? Eine Möglichkeit besteht im Herauskristallisieren elementarer Konstruktionsgesetze aus vielen Anwendungen und ihrer systematischen Darstellung. Aber Programmieren ist ein weites und vielfältiges Gebiet, das oft komplexe geistige Tätigkeit erfordert. Die Vorstellung, es je zu einem reinen "Unterricht von Rezepten" zusammenfassen zu können, ist verfehlt. Aus unserem Arsenal von Unterrichtsmethoden bleibt nur die sorgfältige Auswahl und Darstellung von Musterbeispielen. Natürlich sollten wir nicht glauben, daß jedermann gleich viel aus dem Studium von Beispielen lernt. Wesentlich bei diesem Vorgehen ist, daß vieles dem Studenten, seinem Fleiß und seiner Intution überlassen bleiben muß. Dies gilt ganz besonders für relativ schwierige und lange Programm-

beispiele. Ihr Einschluß in dieses Buch geschah nicht zufällig. Längere Programme sind in der Praxis der "Normalfall" und eignen sich eher dafür, die schwer zu fassende aber wesentliche Zutat, genannt Stil und methodische Struktur, aufzuzeigen."

Besonderer Wert wird auf eine für kommerzielle Anwendungen geeignete Programmierung gelegt. Schwerpunkte sind deshalb:

- Abprüfen von Eingabedaten
- Korrektur von Eingabedaten
- Feldverarbeitung
- Änderbarkeit von Programmen
- Laufzeitreduzierung
- Speichernutzung

Damit ein guter Programmstil geschult wird, erfolgt die Programmierung unter Nutzung der Strukturierungsmöglichkeiten von BASIC.

Das Buch geht möglichst wenig auf die Maschinenarchitektur ein. Deshalb wird Graphik und der Befehl VARPTR nicht behandelt. Die einzige engere Verbindung zur Maschinenarchitektur, die in diesem Buch geschaffen wird, ist das Einlesen des Bildschirmspeichers durch PEEK-Befehle. Diese Operation ist praktisch bei allen Tischcomputern anwendbar, bei denen der Bildschirmspeicher über den Adressbus direkt adressierbar ist.

Das Buch enthält auch einige wertende Urteile. Ich hatte mir lange überlegt, ob ich solche Urteile ins Buch aufnehmen soll oder nicht. Da auch ein Fachautor nicht meinungslos sein kann, sind sie enthalten. Sie sind ausnahmslos durch Rahmung besonders gekennzeichnet.

R.-Fr. Matthaei

Praktische Hinweise

Praktische Hinweise

Schreibmaschinenkenntnisse:
Erleichtern das Durcharbeiten wesentlich.

Speichern von Programmen auf Kassetten:
Bereitet regelmäßig Probleme. Dieses Buch behandelt dieses Thema absichtlich nicht, denn Kassettenbetrieb ist in einer kommerziellen Umgebung undenkbar.

Bildschirmergonomie:
Bei den geringsten Schwierigkeiten mit den Augen sollte man sofort die Arbeit am Bildschirm unterbrechen. Es helfen kürzere Arbeitsperioden (auch ein Besuch beim Augenarzt ist in diesem Fall zu empfehlen).

Ein Befehl pro Zeile:
Ist für Anfänger übersichtlicher als viele Befehle in einer Zeile. BASIC ist jedoch eine Sprache, die zeilenweise und nicht befehlsweise verstanden sein will. Außerdem ist bei Beschränkung auf einen Befehl pro Zeile eine gute Strukturierung des Programmes nicht möglich.

Komplizierte Fallstudien:
Die Fallstudien sind von der Logik her nicht ganz einfach. Man kann sich den Ablauf jederzeit verständlich machen, indem man selbst "Computer spielt" und auf einem Blatt Papier die Inhalte der Variablenfelder bei einem Programmablauf mit Hilfe von Testdaten verfolgt.

Debugging (Fehlersuche):
Dazu gibt es als Hilfsmittel die Befehle TRON, TROFF, STOP und CONT sowie den am bei der 1. Lektion behandelten Befehlsmodus.

Deutsche Fachzeitschriften:
Der Autor hält für empfehlenswert: ELCOMP (Hofacker Verlag) und MC (Franzis Verlag).

1. Lektion

1. LEKTION

BASIC ist ein Kunstwort. Es setzt sich zusammen aus den Anfangsbuchstaben von Beginner's All-Purpose Symbolic Instruction Code (übersetzt: Allzweck-Programmiersprache für Anfänger).
BASIC wurde in den 60er Jahren von John Kemeny und Thomas Kurtz am Dartmouth College (Hannover, New Hampshire, U.S.A.) entwickelt und 1964 erstmals angewandt. BASIC ist ein Nachfolger der in den 50er Jahren eingeführten Programmiersprache FORTRAN. BASIC wurde im Laufe der Jahre erheblich erweitert und präsentiert sich heute als moderne und leistungsfähige Programmiersprache.

Wesentliche Merkmale von BASIC sind:

- BASIC ist extrem einfach zu erlernen.
- BASIC ermöglicht strukturierte Programmierung.
- BASIC ist geeignet für mathematische Applikationen und zur Textbearbeitung.
- BASIC hat eine Schnittstelle zur Assemblerprogrammierung.

Beim Erlernen von BASIC konzentrieren wir uns zunächst auf einen Grundsprachumfang. Später kommt die Stringverarbeitung hinzu.

Überall da, wo sich die amerikanischen Ausdrücke im Sprachgebrauch besser durchgesetzt haben, wenden wir die amerikanischen Ausdrücke an. So sagen wir "Stringverarbeitung" statt "Zeichenkettenverarbeitung".

Der Computer

Unser Tischcomputer Video-Genie oder TRS-80 besteht aus:

- der Tastatur
- dem Bildschirm
- der Zentraleinheit

Er kann erweitert werden durch Peripheriegeräte. Beispiele für Peripheriegeräte sind:

- Kassettenrecorder
- Diskettenlaufwerk
- Drucker

Über die Tastatur werden Kommandos an die Zentraleinheit gegeben. Die Zentraleinheit reagiert auf die Kommandos in genau definierter Weise.

Die Informationsflußrichtung läßt sich wie folgt darstellen:

Es gibt keinen unmittelbaren Informationsfluß von der Tastatur zum Bildschirm, die Verbindung geht immer über die Zentraleinheit.

Die Betriebsarten (Modes)

Befehlsmodus:
Im Befehlsmodus führt der Computer eine auf dem Bildschirm sichtbare Kommandozeile sofort aus.

Der Computer zeigt durch das Zeichen
›
links in einer Zeile, daß er sich im Befehlsmodus befindet.

› wird "Prompt" genannt. Wenn eine Eingabe erwartet wird, befindet sich an der nächsten Einschreibstelle der "Cursor". Der Cursor wird als Unterstreichungszeichen dargestellt.

Ausführungsmodus:
In diesem Modus wird ein BASIC-Programm durchgeführt.

Editiermodus:
Dieser Modus dient der Bequemlichkeit und soll das Ändern von Programmen erleichtern.

Monitormodus:
In diesem Modus werden Nicht-BASIC-Programme bearbeitet. Da wir einen BASIC-Kursus durchführen, interessiert uns hier dieser Modus nur am Rande.

Zum Umschalten von einem Modus in den anderen gibt es Kommandos oder Ereignisse:

Befehlsmodus ⇒ Ausführungsmodus:
Kommandos RUN und GOTO

Ausführungsmodus ⇒ Befehlsmodus
Break-Taste (manueller Programmabbruch)
Programmende (normales Programmende)
Programmfehler (nicht normales Programmende)

Befehlsmodus ⇒ Editiermodus
Kommando EDIT

Editiermodus ⇒ Befehlsmodus
New-Line-Taste
Q-Taste

Ausführungsmodus ⇒ Editiermodus
Programmfehler (automatisch)

Editiermodus ⇒ Ausführungsmodus
kein unmittelbarer Übergang möglich

Die Tastatur im Befehlsmodus

Die Tastatur erinnert an eine Schreibmaschinentastatur. Besonders wichtig sind die Funktionstasten:

New Line:
Beendet die Eingabe einer Zeile. Der Computer interpretiert die Zeile erst nach Betätigen der New-Line-Taste und verhält sich entsprechend dem eingegebenen Kommando. Beim TRS-80 heißt die entsprechende Taste ENTER, beim APPLE II heißt sie RETURN.

Backspace:
Löscht das zuletzt eingetippte Zeichen. Die Backspacetaste ist die Taste mit dem Pfeil nach links.

Shift mit Backspace:
Löscht die zuletzt eingetippte Zeile.

Die Befehlsinterpretation im Befehlsmodus

Der Computer führt eine eingetippte Zeile, die nicht mit einer Ziffer beginnt, sofort aus.
Beginnt die Zeile mit einer Ziffer, so speichert er die Zeile in seinem Speicher ab ohne sie auszuführen.

Da der Computer eine eingetippte Zeile ohne Zeilennummer sofort ausführt, kann man ihn ähnlich wie einen Taschenrechner benutzen wenn man den Befehl
PRINT
verwendet. Statt PRINT kann man auch ein
?
verwenden.

Übung

Schalten Sie den Computer ein und tippen Sie die folgenden Zeilen ein, damit sie sofort ausgeführt werden. Vergleichen Sie die Ergebnisse gegen die Musterergebnisse und vermerken Sie Abweichungen. Korrigieren Sie Falscheingaben mit der Backspace-Taste !

Übung Nr.	Eingabe	Anzeige
1	PRINT 4+8	12
2	? 4*8	32
3	? 4*8+4	36
4	? 4*(8+4)	48
5	? 4=2	0
6	? 4=4	−1
7	? MEM	22123
8	A=190	READY
9	? A	190

10	? MEM	22116
11	? SQR(A)	13.784
12	? SQR(A)*SQR(A)	190
13	?A=A=A	0
14	?A	190
15	A=A=A	READY
16	?A	−1
17	?MEM	22116
18	NEW	READY
19	? MEM	22123
20	? A	0

Anmerkungen zu den Übungen

3 und 4:
Mit Klammern kann die Rangfolge der Operationen verändert werden. BASIC hat von Haus aus folgende Rangfolge innerhalb derselben Stufe (z. B. innerhalb derselben Klammer):

Potenzieren	A[B	([bedeutet: Pfeil aufwärts)
Negieren	−X	
Punktrechnung	*, /	(von links nach rechts)
Strichrechnung	+, −	(von links nach rechts)
logische Operationen mit <, >, =, < =, > =, <>		(von links nach rechts)
NOT		
AND		
OR		

5 und 6:
Die beiden Ausdrücke sind logische Operationen. Wahr wird als −1 dargestellt, unwahr als 0.

7:
Die Pseudovariable MEM enthält die Anzahl der noch freien Speicherplätze in Bytes. Die angezeigte Zahl ist von vielen Faktoren abhängig: Ausbaustufe des Speichers, Betriebssystem, Größe des BASIC-Programmes, ob Variablen eröffnet sind, wie beim Start des Computers die Frage nach der Speichergröße beantwortet wurde, . . .

8, 9 und 10:
Die numerische Variable mit dem Namen A wird eröffnet. Sie belegt 7 Bytes im Speicher.

11:
SQR ist eine mathematische Funktion. Die Klammerung gehört zur Syntax und ist somit vorgeschrieben. Weitere mathematische Funktionen sind z. B:

SIN, COS, TAN	Winkelfunktionen
LOG	natürlicher Logarithmus
EXP	Umkehrung von LOG

13:
Diese logische Funktion ist unwahr. Die Auflösung erfolgt von links nach rechts. Das erste Teilergebnis aus A=A ist wahr, also −1. Die Behauptung −1=A ist jedoch unwahr, da A den Wert 190 hat.

15:
Hier wird der Variablen A das Ergebnis der Behauptung A=A zugewiesen. Diese Behauptung ist wahr, denn 190 ist gleich 190.

16 bis 20:
Das Kommando NEW löscht den Speicher, wie man an den beiden MEM-Anzeigen erkennen kann. Außerdem werden die Variablen geschlossen.

20:
Eine nicht eröffnete numerische Variable hat den Wert 0 und benötigt keinen Speicherplatz. Eine einmal eröffnete numerische Variable benötigt jedoch auch dann Speicherplatz, wenn sie den Wert 0 hat.

Übung zu FOR-TO-NEXT

Wir wollen den PRINT-Befehl genauer untersuchen. Dazu benötigen wir jedoch zuerst etwas Theorie und lernen die Grundlagen der FOR-TO-NEXT-Schleife kennen. Wir erfahren auch, daß man in einer einzigen Zeile durch Verwendung von Doppelpunkten mehrere Kommandos eingeben kann.

Die allgemeine Schreibweise der FOR-TO-NEXT-Schleife ist:
FOR Laufvariable = Anfangswert TO Endwert STEP Schrittweite

Dabei sind FOR, TO und STEP dem BASIC bekannte Schlüsselworte. Die Zeile FOR TO eröffnet eine Wiederholungsschleife in deren Verlauf eine Folge von Programmanweisungen mehrfach ausgeführt werden kann. Das Schlüßelwort NEXT zeigt das Ende der Wiederholungsschleife an.

Übung:
Geben Sie im Befehlsmodus ein:
FOR A=1 TO 100 STEP 10: ?A: NEXT

Ergebnis:
Es werden die Zahlen 1, 11, 21, . . . 91 ausgegeben.

Die Schleife arbeitet so:
Wenn die FOR-Anweisung erstmalig ausgeführt wird, so wird die Laufvariable auf den Anfangswert (hier:1) gesetzt. Die Ausführung läuft bis zur NEXT-Anweisung. In der NEXT-Anweisung wird der bei STEP angegebene Wert zur Schleifenvariablen hinzuaddiert. Wenn STEP nicht angegeben ist, wird der Wert +1 hinzuaddiert. Es wird nach der Addition der Zähler mit dem Endwert verglichen der in der FOR-Anweisung festgelegt wurde. Abhängig vom Wert des Zählers erfolgt eine Wiederholung der Schleife oder nicht.

Übungen:
Lassen Sie im Befehlsmodus folgende Programme laufen und beobachten Sie dabei die angezeigten Werte:

FOR A=100 TO 1 STEP -10: ?A: NEXT
FOR A=-10 TO 100: ?A: NEXT
FOR A=-10 TO 100: ?A,: NEXT
FOR A=-10 TO 100: ?A;: NEXT

Beobachtungen:

- Beim PRINT-Befehl unterdrückt ein Komma oder ein Semikolon nach dem Variablennamen den Zeilenvorschub.
- Beim Komma wird tabelliert, beim Semikolon nicht.
- Das Minuszeichen steht vor der Zahl. Es wird bei der Ausgabe immer

mindestens ein Zwischenraum (Space) zwischen den Zahlen gelassen (diese Aussage gilt nur wenn eine numerische Variable auf eine numerische Variable folgt. Sie gilt nicht für Stringvariable. Diesen Variablentyp werden wir später genau kennenlernen).

– Bei NEXT braucht die Schleifenvariable nicht mit angegeben zu werden, sofern eine nicht verschachtelte Schleife durchgeführt wird.

Übung zur Laufzeitermittlung

Wir wollen untersuchen, ob die Schreibweise einen Einfluß auf die Laufzeit hat. Zu diesem Zweck lassen wir eine Schleife laufen in der nichts getan wird. Der Endwert der Schleife wird so gewählt, daß einige Sekunden Laufzeit entstehen.

```
FOR A=1 TO 10000:NEXT A          (Laufzeit: 30s)
FOR A=1 TO 10000:NEXT            (Laufzeit: 24s)
FORA=1TO10000:NEXT               (Laufzeit: 23s)
FORA%=1TO10000:NEXT              (Laufzeit: 14s)
```

Diskussion der Ergebnisse:
NEXT ohne Angabe der Laufvariablen beschleunigt deutlich.
Das Weglassen der Spaces beschleunigt unwesentlich.
Ein %-Zeichen hinter dem Namen der Laufvariablen beschleunigt sehr deutlich.

2. Lektion

2. LEKTION

Variablennamen und Felder

BASIC kennt "globale" und "lokale" Variable. Globale Variable sind für das gesamte Programm gültig, lokale Variable sind nur für einen bestimmten Programmabschnitt gültig. Beim Video-Genie und beim TRS-80 ohne Diskettenbetriebssystem werden lediglich globale Variable unterstützt.

Alle Variablenamen müßen mit einem Buchstaben beginnen. Ein weiterer Buchstabe oder eine Ziffer kann folgen.

Bei Variablennamen mit einer Länge von mehr als 2 Zeichen werden nur die beiden ersten Zeichen von BASIC als Auswahlkriterium betrachtet.

Beispiele:

A B2 C DA GH ABO AB

Alle Namen sind zulässig. ABO und AB kennzeichnen jedoch die gleiche Variable.

BASIC-Schlüsselworte dürfen nicht als Variablennamen verwendet werden, da sie falsch interpretiert werden. So wird z. B. der nicht zulässige Variablenname TOT als Schlüßelwort TO mit nachfolgendem Variablennamen T interpretiert.

Ein Suffix hinter dem Variablennamen gibt den Typ der Variablen an:

Suffix	Bereich	Beispiel
%	ganzzahlig numerisch −32768 bis 32767	A%
!	einfach genau, numerisch 6 gültige Stellen	A!
#	doppelt genau, numerisch 16 gültige Stellen	A#
$	Stringvariable, alle Zeichen	A$

Sofern kein Suffix angegeben wird, wird der Suffix ! unterstellt. A%, A!, A# und A$ kennzeichnen verschiedene Variablen, die alle den Namen A haben.

Übung

Im Befehlsmodus:

A%=1:A!=2:A#=3:A$="4":?A%, A!, A#, A$, A

Ergebnis: 1 2 3 4 2

Die Standardzuordnung kein Suffix = ! kann verändert werden durch die Kommandos:

DEFDBL	für doppelte Genauigkeit
DEFINT	für ganzzahlig
DEFSNG	für einfache Genauigkeit
DEFSTR	für String

So bedeutet DEFINT A–C,Z , daß alle Variablennamen, die mit A,B,C oder Z beginnen und bei denen kein Suffix angegeben ist, als ganzzahlige Variablen zu betrachten sind.

Übung (im Befehlsmodus):

```
DEFINT I,J:I=10:J=20:K=30:?I, I%, J, J%, K, K%

Ergebnis:  10        10        20        20        30        0
```

Übung zur Typenvereinbarung und PRINT:
(im Befehlsmodus):

```
NEW
I%=10:J%=3:?I%/J%          Ergebnis: 3.33333
K%=I%/J%:?K%               Ergebnis: 3
K#=I%/J%:?K#               Ergebnis: 3.333333253860474
```

An dieser Stelle sei ausdrücklich darauf hingewiesen, daß die mathematischen Funktionen mit einfacher Genauigkeit berechnet werden. Die Spezifikation von doppelt genauen Variablentypen im Zusammenhang mit mathematischen Funktionen wie SQR, COS, SIN, LOG usw. bringt keine Verbesserung der Rechengenauigkeit.

Felder

Jeder Variablenname kann als Feld benutzt werden. Ein Feld ist eine geordnete Liste von Werten, wobei ein Index die Position innerhalb der Liste angibt. Der Index kann ein- oder mehrstufig sein. Man spricht auch von ein- und mehrdimensionalen Feldern.

Wir betrachten zunächst nur die eindimensionalen Felder:

Eindimensionale Felder

Eindimensionale Felder müssen mit DIM deklariert werden, sofern andere Indexwerte als 0 bis 10 verwendet werden sollen.

DIM A(30) deklariert das Feld A mit den Positionen A(0) bis A(30).

Übung:
Ermitteln Sie im Direktmodus den Speicherbedarf der durch die

Deklaration mit DIM bei verschiedenen Variablentypen belegt wird:

NEW	NEW	NEW	NEW
?MEM	?MEM	?MEM	?MEM
DIM A(30)	DIM A%(30)	DIM A#(30)	DIM A$(30)
?MEM	?MEM	?MEM	?MEM
Ergebnis:			
132 Bytes	70 Bytes	256 Bytes	101 Bytes

Nach der Deklaration mit DIM haben die Feldinhalte numerischer Variablen den Inhalt 0 (Stringvariablen sind nach der Deklaration mit DIM "leer").

Übung (im Befehlsmodus):

```
DIM A(40): FOR I=0 TO 40:?A(I);:NEXT
```

Erst durch Zuweisen erhalten die Elemente des Feldes einen Wert, z. B. so:

```
A(21)=10:A(22)=15
FOR I=0 TO 40:?A(I);:NEXT
```

Schöner wäre es, wenn neben der Anzeige des Inhaltes auch der Index stehen würde. Dies kann man erreichen durch Erweiterung des PRINT-Befehles mit der String "=":

```
FOR I=0 TO 40:?I;"=";A(I);:NEXT
```

Eindimensionale Felder werden zusammen mit FOR-TO-NEXT gerne verwendet, um Eingabewerte rationell bearbeiten zu können und um sie nachträglich korrigieren zu können.

Dieses Thema nutzen wir für unsere erste

Fallstudie

Wir werden eine Lösung erarbeiten, die es ermöglicht, mehrere Zahlen einzugeben, die eingegebenen Zahlen zu überprüfen und bei fehlerhafter Ersteingabe nachträglich zu korrigieren.

Anschließend wird der Durchschnitt aus den eingegebenen Zahlen ermittelt.

Theoretischer Unterbau:
Das Programm wird so lang werden, daß es sich in einer Zeile nicht mehr programmieren läßt. Deshalb muß es im Ausführungsmodus ablaufen. Wir werden dazu die Programmzeile abspeichern lassen, indem wir jede Zeile mit einer Zeilennummer beginnen lassen.

Mit dem Kommando

LIST

den wir im Befehlsmodus jederzeit eingeben können, haben wir die Möglichkeit die abgespeicherten Programmzeilen auf dem Bildschirm erneut sichtbar zu machen.

Wir wissen noch nicht, wie man eine Eingabe über die Tastatur codiert. Deshalb programmieren wir zuerst nur den Teil des Programmes, der den Durchschnittswert errechnet. Die Wertzuweisungen zu den Elementen des Feldes erfolgen durch eine FOR-TO-NEXT-Schleife.

Geben Sie bitte folgendes Programm ein:

```
10 REM DURCHSCHNITT ERRECHNEN
20 DIM A(50):REM FUER MAX. 51 EINGABEWERTE
200 FOR I=0 TO 50:A(I)=I:NEXT :REM A(0)=0, A(
15)=15,USW.
210 FOR I=0 TO 50:S=A(I)+S:NEXT :REM SUMMIEREN
220 ?"SUMME=";S
230 ?"DURCHSCHNITT=";S/I
```

Diskussion

Bei Zeile 230 wird durch I dividiert, nicht durch 51. Hätte man codiert: 230 ?"DURCHSCHNITT=";S/51 so hätte man bei jeder Änderung der Anzahl der Werte die Zeile 230 ändern müssen.
Durch Ergänzen der Zeile 15 DEFINT I wird das Programm schneller laufen.
Mit dem Kommando RUN (im Befehlsmodus) wird die Ausführung des Programmes gestartet.

Nach Ende des Programmes kann man im Befehlsmodus die eröffneten und deklarierten Variablen überprüfen, z. B. mit

```
?A(22)          22
?I              51
```

Durch jede Veränderung des Programmes werden die Deklarationen aufgehoben und die Variablen geschlossen.
Beispiel: Erweitern Sie das Programm um die Zeile 240 END.

```
240 END
?A(22)          bringt eine Fehlermeldung
?I              0
```

END

Die Zeile 240 zeigt das Programmende an. Der Befehl hat an dieser Stelle lediglich Kommentararcharakter, da er gleichzeitig letzter Befehl des Programmes und somit entbehrlich ist. Es gibt Fälle, wo das Ende des Programmablaufes nicht auch der letzte Befehl des Programmes ist. An dieser Stelle ist die Verwendung des Befehles END angebracht. Der Befehl END bewirkt eine Umschaltung vom Ausführungs- in den Befehlsmodus.

RUN

Der Befehl RUN bewirkt außer dem Übergang vom Befehls- in den Ausführungsmodus auch eine Schließung der Variablen. Genau aus diesem Grund braucht die Variable S in dem Programm nicht gelöscht (d. h., auf 0 gesetzt) zu werden: geschloßene numerische Variable haben den Wert 0.

3. Lektion

3. LEKTION

Die nächste logische Entwicklungsstufe des Fallstudienprogrammes ermöglicht die Eingabe über Tastatur:

```
10 DEFINT I,K:J=50
15 DIM A(J)
20 CLS:PRINT"DURCHSCHNITT ERRECHNEN, MAXIMAL"
;J;" EINGABEWERTE"
30 PRINT"EINGABE DER WERTE. SCHLUSS MARKIEREN
DURCH EINGABE VON -999"
40 FOR I=1 TO J
50 PRINT I;". WERT=";:INPUT A(I)
60 IF A(I)=-999 THEN K=I-1:I=J+1:REM SCHLEIFEN
AUSSTIEG, WENN FERTIG
70 NEXT
210 FOR I=1 TO K:S=A(I)+S:NEXT
220 PRINT"SUMME=";S
230 PRINT"DURCHSCHNITT=";S/(I-1)
240 END
```

Diskussion des Programmes:
Zeile 10 bewirkt, daß Variablennamen, die keinen Suffix besitzen und die mit I, J oder K beginnen, vom Typ "Ganzzahlig numerisch" sind. Durch die möglichst weitgehende Verwendung von ganzzahligen numerischen Variablen wird der Programmablauf beschleunigt.

Zeile 15 deklariert maximal 51 Positionen für das Feld A.

Zeile 20 löscht mit dem Befehl CLS den Bildschirminhalt. Hier ist eine Anmerkung angebracht:
Als in den 60er Jahren BASIC entwickelt wurde, waren Bildschirmsichtgeräte unbekannt. Daraus ergibt sich, daß die im ursprünglichen BASIC vorgesehene Vorschubsteuerung für Fernschreiber konzipiert war. Somit muß man davon ausgehen, daß in unterschiedlichen BASIC-Versionen Kommandos zur Vorschubsteuerung auf dem Bildschirm ebenfalls unterschiedlich sind. CLS ist typisch hierfür: bei anderen Systemen wird der Bildschirm mit beispielsweise HOME, CALL CLEAR oder mit Ausgabe des ASCII-Steuerzeichens für Blattvorschub (ASCII bhandeln wir später) gelöscht.

Die in Zeile 20 und 30 programmierten PRINT-Befehle kennzeichnen das Programm gegenüber dem Benutzer und geben wichtige Bedingungshinweise. Auch hier sind zwei Informationen angebracht:
Man kann 3 Bytes an Programmlänge einsparen indem man codiert:

```
20 CLS:PRINT"DURCHSCHNITT ERRECHNEN, MAXIMAL"J;
" EINGABEWERTE
30 PRINT"EINGABE DER WERTE. SCHLUSS MARKIEREN
DURCH EINGABE VON -999
```

Im Unterschied zu den im Programm codierten Zeilen 20 und 30 fehlen die abschließenden " und vor dem Variablennamen das ; .

Beim BASIC des TRS-80 und Video Genie kann man Zeile 20 und 30 unter einer einzigen Zeilennummer abspeichern. Dadurch werden 4 Bytes Speicherplatz gespart und die Übersichtlichkeit des Programmes erhöht (die Taste Pfeil abwärts muß vor dem Wort "SCHLUSS" betätigt werden):

```
20 CLS:PRINT"DURCHSCHNITT ERRECHNEN, MAXIMAL"J
;"EINGABEWERTE
SCHLUSS MARKIEREN DURCH EINGABE VON -999
```

Zeilen 40 bis 70:
Hier erfolgt die Eingabe der Werte innerhalb der FOR-TO-Schleife.

Dabei bewirkt Zeile 50 folgende Darstellung auf dem Bildschirm:

20 . WERT=?

Das Space zwischen der Anzeiger der Laufvariablen und dem Punkt stört etwas. Wir werden später lernen, wie es sich durch Anwendung der USING-Anweisung vermeiden läßt. Das Fragezeichen wird durch den Eingabebefehl INPUT erzeugt. In der hier codierten Form verlangt INPUT die Eingabe einer Zahl die es in der Feldvariablen A (Laufvariable) abspeichert.

Die Schleife beginnt bei 1. Dadurch wir A(0) nicht genutzt. Der Grund ist, daß wir die Anzeige

0 . WERT=?

vermeiden wollen.

Die Eingabe von –999 bewirkt eine vorzeitige Beendigung der FOR-TO-Schleife. Die IF-Abfrage in Zeile 60 hat das Ergebnis "wahr" (interne Darstellung als –1) bei Eingabe von –999. Nur in diesem Fall wird der Teil der Zeile hinter dem Schlüsselwort THEN ausgeführt. (Die Abgrenzung ab wo die Programmausführung bei "unwahr" fortgesetzt wird ist bei verschiedenen BASIC-Dialekten nicht einheitlich definiert. Die genaue Abgrenzung für den BASIC-Dialekt des Video-Genies oder TRS-80 wird in der nächsten Lektion genau geschrieben.)

Hinter THEN erfolgt der Reihe nach:

K=I–1	K enthält den Index des letzten gültigen Eingabewertes. Das ist der Wert, der vor –999 eingetippt wurde.
I=J+1	I enthält nun die Summe aus Endwert + Schrittweite. Das bedeutet, daß bei der NEXT-Ausführung die Schleife nicht wiederholt wird.

Beachten Sie: es ist bei den BASIC-Dialekten verschiedener Homecomputer (der TRS-80 und der Video Genie gehören hierzu) nicht zulässig, eine FOR-TO-Schleife anders als über den NEXT-Befehl zu verlassen. Die in diesem Programm praktizierte Art des Schleifenausstieges ist die einzig zulässige Art bei diesen Geräten.

Zeile 210 bis 240 kennen wir bereits. Geändert hat sich lediglich in Zeile 210 der Endwert: er wird jetzt in Abhängigkeit von der Anzahl mitgegebener Werte bestimmt. In Zeile 230 wird durch I−1 dividiert, da A(0) nicht mehr mitverwendet wird.

Überblick

Ist unser Programm jetzt endlich fertig ?
Anwort: Nein, es fehlt:
Plausibilierung der Eingabewerte
Korrekturmöglichkeit für Eingabewerte nach erfolgter Eingabe.

Bei wechem Thema sind wir überhaupt ?
Antwort: Wir behandeln eine Fallstudie zur Verarbeitung eindimensionaler Felder.

Wie geht es denn weiter ?
Antwort:
Wir perfektionieren unser Fallstudienprogramm, behandeln dabei die IF-THEN-ELSE-Abfrage und Unterprogramme. Anschließend beginnen wir mit der Besprechung zweidimensionaler Felder.

Bevor wir die Fallstudie fortsetzen, lernen wir einen weiteren Befehl kennen:

GOTO Zeilennummer

GOTO übergibt die Programmsteuerung an den Beginn der angegebenen Zeile.
Wird GOTO alleine verwendet, so handelt es sich um eine unbedingte

Verzweigung. Diese sollte in Programmen vermieden werden, da sie die Übersichtlichkeit reduziert.
Vor GOTO können in der gleichen Befehlszeile Abfragen stehen. Man spricht dann von einer bedingten Verzweigung.

GOTO Zeilennr. kann auch im Befehlsmodus eingegeben werden. Es wirkt wie RUN Zeilennr., schliesst jedoch im Gegensatz zu RUN und RUN Zeilennr. die Variablen und Deklarationen nicht. GOTO Zeilennr. ist dadurch ein wichtiges Hilfsmittel bei der Fehlersuche in Programmen.

Übung (im Befehlsmodus):

```
1 A=20
2 ?"A="A
3 STOP
```

Starten Sie das Programm nacheinander mit:

Startkommando	Ergebnis
RUN	A= 20
RUN 2	A= 0
RUN	A= 20
GOTO 2	A= 20

Fortsetzung der 1. Fallstudie

In der Fallstudie fehlt noch die Plausibilierung der Eingabedaten. Sämtliche Eingabedaten für ein Computerprogramm müssen plausibiliert werden, denn Computer bearbeiten ohne weiteres falsche und richtige Daten. Die Menschheit verdankt der mangelnden Plausibilierung von Eingabedaten in Computersysteme eine Vielzahl der sogenannten "Computerwitze".

Wenn wir uns vorstellen, daß die eingegebenen Werte in unserem Fallstudienprogramm die Körperlänge erwachsener Menschen sein soll, so können wir daraus Plausibilierungsanforderungen ableiten. Bevor wir das machen, noch ein Hinweis: in Fachzeitschriften und Fachbüchern findet man häufig kleinere BASIC-Programme aus verschiedenen Bereichen abgedruckt. Bevor man ein solches Programm in die Praxis für eine Anwendung übernehmen kann, muß man die Plausi-

bilierung der Eingabedaten ergänzen.

Bezogen auf unsere Daten können wir sagen:

1. Es gibt keinen Erwachsenen der kleiner ist als 0.7 m und der größer ist als 2.5 m.

Somit können wir Zeile 62 ergänzen:

```
62 IF A(I)<.7 OR A(I)>2.5 THEN ?"NUR 0.7 BIS
2.5 ZULAESSIG.
NEUEINGABE !":GOTO 50
```

Allerdings folgt nun die Zeile 70 nicht mehr unmittelbar auf die Zeile 60. Deshalb müssen wir vor REM in die Zeile 60 einfügen:

GOTO 70:

Die Änderungen bewirken, daß bei Eingabe von unzulässigen Werten ein Fehlerhinweis auf dem Bildschirm angezeigt wird und daß eine Wiederholung der Eingabe verlangt wird, indem mit einem bedingten GOTO 50 zur Eingabe zurückverzweigt wird.

Man kann noch sagen:

2. Normalerweise sind Erwachsene zwischen 1.45 und 2.20 m groß. Es ist nicht häufig, daß ein Erwachsener zwischen 0.7 und 1.45 oder zwischen 2.20 und 2.50 groß ist. Werden Werte in diesem Bereich eingegeben, so soll das Programm rückfragen, ob die Eingabe korrekt war. Die Anwort soll "N" oder "J" sein.

Die zu ergänzenden Zeilen lauten:

```
64 IF A(I)<1.45 OR A(I)>2.2 THEN "INPUT"KORREK
T (J/N)";A$:
IF A$<>"J" AND A$<>"N" GOTO 64
66 IF A$="N" GOTO 50:REM NEUEINGABE WENN NICHT
KORREKT
```

Zeile 64 engt den Bereich erneut ein. Das Programm fragt zurück, ob die Eingabe korrekt war. Auch die Antwort (Taste J oder N) wird vor ihrer Auswertung plausibiliert: unzulässige Eingaben verarbeiten wir auf gar keinen Fall !

Unser kleines Fallstudienprogramm sieht jetzt so aus – es wurde noch eine kleine Änderung in Zeile 50 hineingebracht, die eine Verbesserung der Anzeige bewirkt. Vor der Änderung wurde gezeigt:

20 . WERT=? nun wird gezeigt: 20. WERT=?
(Wie USING funktioniert erfahren wir später . . .)

```
10 DEFINT I,K:J=50
15 DIM A(J)
20 CLS:PRINT"DURCHSCHNITT ERRECHNEN, MAXIMAL"
;J;" EINGABEWERTE"
30 PRINT"EINGABE DER WERTE. SCHLUSS MARKIEREN
DURCH EINGABE VON -999"
40 FOR I=1 TO J
50 PRINT USING"##."; I;:PRINT" WERT=";:INPUT
A(I)
60 IF A(I)=-999 THEN K=I-1:I=J+1:GOTO 70:REM
SCHLEIFENAUSSTIEG, WENN FERTIG
62 IF A(I)<.7 OR A(I)>2.5 THEN PRINT"NUR .7
BIS 2.5 ZULAESSIG.
NEUEINGABE !":GOTO 50
64 IF A(I)<1.45 OR A(I)>2.2 THEN INPUT "KORRE
CKT (J/N)";A$:
IF A$<>"J" AND A$<>"N" GOTO 64
66 IF A$="N"THEN A$="":GOTO 50:REM NEUEINGABE
70 NEXT
210 FOR I=1 TO K:S=A(I)+S:NEXT
220 PRINT"SUMME=";S
230 PRINT"DURCHSCHNITT=";S/(I-1)
240 END
```

NOTIZEN

4. Lektion

4. LEKTION

IF

In dem Fallstudienprogramm haben wir den Befehl IF verwendet. Es ist an der Zeit, ihn auch ordentlich zu erklären:
Die allgemeine Schreibweise von IF ist:

IF Logischer Ausdruck Ausführungsteil
Zwischen dem logischen Ausdruck und dem Ausführungsteil kann zur Trennung das Schlüsselwort THEN verwendet werden.

Der logische Ausdruck hat als Ergebnis wahr oder unwahr. Von der 1. Lektion her wissen wir, daß wahr als −1 und unwahr als 0 dargestellt wird. Diese Darstellung ist jedoch nicht allgemein üblich und es ist schlechter Programmierstil, die interne Darstellung beispielsweise wie folgt auszunutzen:

IF A wirkt wie IF A<>0
IF A OR B wirkt wie IF A<>0 OR B<>0.

Der logische Ausdruck kann mit AND, OR oder NOT verknüpft werden. In der Fallstudie haben wir die AND- (Und-) Verknüpfung verwendet.

THEN ohne ELSE

Der Ausführungsteil kann mit dem Schlüsselwort THEN eingeleitet werden. Häufig ist die Verwendung dieses Schlüsselwortes wahlfrei: In Zeile 62 und 64 kann das THEN weggelassen werden. Man spart

dadurch Speicherplatz.

Merken Sie sich jedoch folgendes:

Eine Zeilennummer darf niemals alleine stehen. Das bedeutet, wenn THEN an einer Stelle steht wo es gegen GOTO ersetzt werden kann, so muß entweder THEN oder GOTO vor der angegebenen Zeilennummer stehen.

> Sie sollen sich bitte nicht dadurch täuschen lassen, daß ein Programm auch häufig bei Nichteinhaltung der eben genannten Vorschrift funktioniert. Weglassen des Wortes THEN oder GOTO vor einer Zeilennummer ist ein Verstoß gegen die Syntax und hat zur Folge, daß Hilfsprogramme wie Renumber (dieses Programm ändert die Zeilennummern eines BASIC-Programmes) nicht mehr einwandfrei funktionieren.

Bei den BASIC-Dialekten des Video-Genie oder des TRS-80 gilt für THEN ohne ELSE in einer Zeile:
Ist die Bedingung wahr, wird der Ausführungsteil in der gleichen Zeile ausgeführt. Ist die Bedingung unwahr, wird der Ausführungsteil in der gleichen Zeile nicht ausgeführt, das Programm wird mit Beginn der nächsten Zeile fortgesetzt.

Bei anderen BASIC-Dialekten kann der Wiederaufsetzpunkt für unwahr anders definiert sein: er kann in der gleichen Zeile liegen wie das Wort THEN selbst.

THEN mit ELSE

Die Verwendung der ELSE-Klausel ermöglicht es, in unserem Falstudienprogramm die Zeile 66 einzusparen indem wir codieren:

```
64 IF A(I)<1.45 OR A(I)>2.2 INPUT"KORREKT (J/
N)";A$:IF A$="N" THEN 50 ELSE IF A$<>"J" THEN
64
```

Sofern in einer Zeile hinter THEN das Schlüsselwort ELSE aufgeführt

ist, wird hinter ELSE ein Ausführungsteil für den Fall ausgeführt, daß die Bedingung unwahr ist.

THEN und ELSE können verschachtelt werden. Wie man an der eben codierten Zeile 64 sieht, sollte man die Verschachtelung nicht zu weit treiben, da sonst leicht die Übersichtlichkeit verloren geht.

Die ELSE-Klausel ist eine wesentliche Voraussetzung für die Erstellung gut strukturierter Programme, denn es ist mit ihrer Hilfe möglich, GOTO-Befehle und Zeilennummern zu vermeiden. Es ist bedauerlich, daß die BASIC-Dialekte einiger Home-Computer die ELSE-Klausel nicht unterstützen. Ohne ELSE sind zum Beispiel: Apple II, Atari 400, Atari 800, Commodore CBM, ITT 2020.
Mit ELSE sind: Video Genie, TRS-80, Texas Instruments TI 99/4.

Korrekturmöglichkeit beim Fallstudienprogramm

Es ist möglich, nach Beendigung der Eingabe durch –999 die eingegebenen Werte anzuzeigen, um sie korrigieren zu können. Bei nur 50 Werten paßt die Anzeige bequem in eine einzige Bildschirmseite, so daß wir sehr einfach programmieren können:

```
80 CLS:FOR I =1 TO K:PRINT USING"##.";  I;:PRI
NT" WERT="A(I),:NEXT:REM KONTROLLANZEIGE
90 INPUT"ALLES OK (J/N)";A$:IF A$="J"THEN 210
ELSE IF A$<>"N"THEN 90
100 INPUT"WERT-NR,NEUER WERT";W,X
110 IF W>K OR W<1 THEN PRINT"NUR WERT-NR.1 BI
S"K;"ZULAESSIG.
NEUEINGABE !":GOTO 90
120 IF X<.7 OR X>2.5 THEN PRINT"UNZULAESSIGER
WERT. NEUEINGABE !":
GOTO 90 ELSE A(W)=X:GOTO 80
```

Diskussion

So ganz korrekt ist unser Programm noch nicht. Wenn wir es laufen lassen und als einzige Eingabe den Wert –999 eingeben, so ist das

Ergebnis noch verkehrt. Mit diesem Problem müssen wir uns noch auseinandersetzen.
Es ist ganz selbstverständlich, daß das Programm auch die bei der Korrektur eingetippten neuen Werte nach den gleichen Regeln plausibilieren muß, wie es bei der Ersteingabe geschehen ist.
Hier zeigt sich eine Schwachstelle unseres bisherigen Programmes: Wir haben die gleiche Plausibilierung zweimal – in Zeile 62 und in Zeile 120 – codiert.
Man sollte in einem Programm grundsätzlich den gleichen Sachverhalt auch dann nur einmal codieren, wenn er mehrfach genutzt wird. Diese Art der Programmierung erleichtert die Erstellung des Programmes und vereinfacht seine spätere Änderbarkeit.

Die Lösung heißt in unserem Fall:
Zeile 62 und 64 werden als Unterprogramm codiert. Dadurch kann die in diesen beiden Zeilen enthaltenen Plausibilierung von mehreren Stellen des Programmes aus genutzt werden. Bevor wir die Änderung durchführen, müssen wir zwei weitere Befehle erlernen, die eine Struktur "Unterprogramm" ermöglichen:

GOSUB und RETURN

GOSUB verzweigt – ähnlich wie GOTO – zur angegebenen Zeilennummer. Der nächste ausgeführte RETURN-Befehl führt eine Rückverzweigung unmittelbar hinter den zuletzt ausgeführten GOSUB-Befehl durch.

Beachten Sie:
GOSUB- und RETURN-Befehle müssen paarig ausgeführt werden.

Eine Verschachtelung ist möglich. Die maximale Verschachtelungstiefe wird beim Video-Genie und beim TRS-80 durch den noch freien Speicherraum bestimmt.

1. Übung zu GOSUB:

Das für diese und die folgende Übung benötigte Testprogramm liegt gemeinsam mit dem Fallstudienprogramm im Speicher und beginnt mit der Zeilennummer 1000. Die beiden Übungsprogramme werden deshalb

mit RUN 1000 gestartet. Sie lassen sich später mit dem Kommando

DELETE 1000–2000

aus dem Speicher löschen.

```
1000 ?"ZEILE 1000":GOSUB 2000
1010 ?"ZEILE 1010
1020 ?"ZEILE 1020":GOSUB 2000
1030 ?"ZEILE 1030
2000 ?"ZEILE 2000":RETURN
```

Dieses Programm beinhaltet einen Fehler. Korrigieren Sie den Fehler und lassen Sie das Programm laufen.

2. Übung zu GOSUB:

Die folgende Programmstruktur ist eigentlich nicht zulässig, da GOSUB und RETURN nicht paarig ausgeführt werden: das Programm beinhaltet keinen einzigen RETURN-Befehl.
Es zeigt die maximal erreichbare Verschachtelungstiefe an:

```
1000 GOSUB 2000
2000 I=I+1:?I;:GOTO 1000
```

Danach im Direktmodus:

?MEM	Anzeige, z. B. 14773
?RUN	letztes I, z. B. 2938
?MEM	Anzeige, z. B. 71

Nach dem Lauf des Programmes ermitteln Sie bitte, ob die bei PRINT MEM gezeigten Werte und die zuletzt gezeigte Zahl I in einem bestimmten Verhältnis stehen. Denken Sie dabei daran, daß Sie den Computer im Befehlsmodus als Taschenrechner verwenden können !

Ergebnis:
Jedes aktive GOSUB belegt offensichtlich 5 Bytes.
Diese Erkenntnis ist interessant, weil in den Handbüchern zum Video-Genie und zum TRS-80 steht: "jedes aktive GOSUB belegt 6 Bytes".

Eine Aufgabe für Sie:
Bitte, stellen Sie die Plausibilierung in dem Fallstudienprogramm so um,

daß Sie sie als Unterprogramm aufrufen können.
Bitte, ändern Sie das Programm so, daß der Fehler bei Eingabe von –999 nicht mehr auftritt.

Musterlösung zur 1. Fallstudie

```
10 DEFINT F,J:J=50
15 DIM A(J)
20 CLS:PRINT"DURCHSCHNITT ERRECHNEN, MAXIMAL"J
;" EINGABEWERTE.
EINGABE DER WERTE, SCHLUSS MARKIEREN DURCH EIN
GABE VON -999
40 FOR I=1 TO J
50 PRINT USING"##."; I;:PRINT" WERT=";:INPUT X
60 IF X=-999 THEN K=I-1:I=J+1:GOTO 70:REM SCHL
EIFENAUSSTIEG, WENN FERTIG
62 GOSUB 300:IF F=1 THEN 50 ELSE A(I)=X:REM PL
AUSI
70 NEXT

75 IF K=0 THEN PRINT"KEINE WERTE EINGEGEBEN  !
":END

77 REM Kontrollanzeige zur Kontrolle
80 CLS:FOR I =1 TO K:PRINT USING"##."; I;:PRIN
T" WERT="A(I),:NEXT:PRINT

90 INPUT"ALLES OK (J/N)";A$:IF A$="J"THEN 210
ELSE IF A$<>"N"THEN 90

100 INPUT"WERT-NR, NEUER WERT";W,X
110 IF W>K OR W<1 THEN PRINT"NUR WERT-NR.1 BIS
"K;"ZULAESSIG.
 NEUEINGABE !":GOTO 90
120 GOSUB 300:IF F=1 THEN 90 ELSE A(W)=X:GOTO
80:REM PLAUSI, GGF.UEBERNEHMEN

209 REM DURCHSCHNITT ERRECHNEN U.PROGRAMMENDE
210 FOR I=1 TO K:S=A(I)+S:NEXT
220 PRINT"SUMME=";S
230 PRINT"DURCHSCHNITT=";S/(I-1)
```

```
240 END

299 REM Plausibilierung der Eingabewerte fuer
die Groesse
    Fehlercode: F=0 wenn o.k., F=1 wenn nicht
o.k.
300 A$="":IF X<.7 OR X>2.5 THEN PRINT"NUR .7
BIS 2.5 ZULAESSIG.
 NEUEINGABE !":GOTO 330
310 IF X<1.45 OR X>2.2 THEN INPUT "KORREKT (J
/N)";A$:IF A$<>"J"
 AND A$<>"N" GOTO 310
320 IF A$="N"THEN 330:REM NEUEINGABE
325 F=0:RETURN:REM Normaler Ausgang
330 F=1:RETURN:REM Fehlerausgang
```

Diskussion der Musterlösung

Äußeres Erscheinungsbild:
Durch Leerzeilen kann man zusammengehörende Programmabschnitte gruppieren und so die Übersichtlichkeit erhöhen.

Zeile 80:
Kleinere zusammenhängende Vorgänge können in einer Zeile codiert werden.

Zeile 299 und 300:
Unterprogramme beginnen grundsätzlich mit einer Kommentarzeile. Diese Kommentarzeile wird nie angesprungen, denn auch die "Ausführung" einer Kommentarzeile verbraucht Rechenzeit.

Zeile 325 und 330:
Das Unterprogramm gibt einen Fehlercode an das Hauptprogramm zurück.

FOR-TO-Problematik:
Bei den BASIC-Dialekten des Video-Genie und des TRS-80 wird eine FOR-TO-Schleife mindestens einmal durchlaufen. Deshalb muß für den Fall, daß sie nicht durchlaufen werden soll, eine Abfrage erfolgen (Zeile 75 im Fall der Musterlösung.)

Es gibt BASIC-Dialekte, bei denen die FOR-TO-Schleife auch ohne die Abfrage in Zeile 75 nicht durchlaufen würde, da der Schleifenendwert bereits beim Eintritt in die Schleife erreicht oder sogar überschritten war. Insbesondere bei Übernahme von BASIC-Programmen von einem System auf's andere kann diese Abweichung in den Dialekten zu Problemen führen.

Übung (im Befehlsmodus):

FOR I=100 TO 99:?I:NEXT

Ergebnis:
Es wird die Zahl 100 auf dem Bildschirm angezeigt, obwohl die Schleifenendbedingung bereits beim Einstieg in die Schleife erfüllt war.

5. Lektion

5. LEKTION

Zwei- und mehrdimensionale Felder

Zwei- und mehrdimensionale Felder müssen auf jeden Fall mit DIM deklariert werden.

Beispiel:

DIM A(20,30),B(10,11,12)

A ist ein zweidimensionales Feld mit 21*31 Elementen, B ist dreidimensional und hat 11*12*13 Elemente.

Drei- und mehrdimensionale Felder treten in kommerziellen Programmen nur selten auf.

Im Bereich der Mathematik jedoch haben sie eine hohe Bedeutung bei der Matrixrechnung. Es sei erwähnt, daß es in komfortablen BASIC-Dialekten eine Befehlsgruppe MAT speziell zur Matrixbearbeitung gibt. Diese Befehlsgruppe fehlt in den Tischcomputern wie APPLE, COMMODORE CBM, Video Genie und TRS-80.

Verschachtelte Schleifen

Zur Bearbeitung mehrdimensionaler Felder sind verschachtelte Schleifen hervorragend geeignet. Dabei wird die innerste Schleife zuerst abgearbeitet, wie die nachfolgende Übung zeigt:

```
10 ?" A","  B","  C"
20 FOR A= 1 TO 2
30 FOR B=11 TO 12
40 FOR C=21 TO 22
50 ?A,B,C
60 NEXT C,B,A
```

Bitte, versuchen Sie, in Zeile 60 die Reihenfolge der Laufvariablen zu verändern.!

In dem nachfolgenden Demonstrationsprogramm "zweidimensionale Matrix" werden verschachtelte Schleifen verwendet. Die Dateneingabe erfolgt über die Befehle DATA und READ:

DATA, READ und RESTORE

DATA ermöglicht es, Daten innerhalb eines Programmes als Bestandteil des Programmes mit anzugeben. Die einzelnen Daten werden durch Kommata getrennt. Alle mit DATA definierten Daten eines Programmes werden sequentiell gelesen, d. h., beginnend mit der ersten DATA-Anweisung des Programmes. DATA-Anweisungen könnnen jedoch an beliebiger Stelle innerhalb des Programmes stehen.

Der Lesebefehl lautet: READ Variablename

Der Befehl RESTORE läßt das Lesen der DATA-Werte erneut beginnen. Bei Ausführung von RUN wird ein RESTORE-Befehl intern mit durchgeführt.

Andere BASIC-Dialekte kennen den Befehl

RESTORE Zeilennummer

Diese Variante ermöglicht ein gezieltes sequentielles Einlesen ab einem bestimmten DATA-Befehl. Diese Befehlsvariante wird vom Video Genie und TRS-80 nicht unterstützt.

Übung:

```
10 READ A$,B,C$,D
20 ?A$,B,C$,D
30 RESTORE
40 READ A$,B$,C$,D$
50 ?A$,B$,C$,D$
60 RESTORE
70 READ A,B,C,D
80 ?A,B,C,D
100 DATA TEDDY,2,"GROSS,KLEIN",4
```

Zeile 70 verursacht eine Fehlermeldung, die sich jedoch auf Zeile 100 bezieht. Korrigieren Sie das Programm entsprechend.

Die Gänsefüsschen bei "GROSS,KLEIN" sind erforderlich, um den String als zusammengehörend zu kennzeichnen. Ohne Gänsefüsschen würde das Komma im String GROSS und KLEIN diesen als zwei getrennte DATA-Werte erscheinen lassen.

Die Art der Daten und der bei READ angegebene Variablentyp muß übereinstimmen. Ist das nicht der Fall, werden die Daten als fehlerhaft gekennzeichnet.

Das folgende Demonstrationsprogramm für zweidimensionale Felder dürfte selbsterklärend sein. Bitte, lassen Sie es einmal laufen !

```
10 REM DEMONSTRATIONSPROGRAMM FUER 2-DIMEN
SIONALE FELDER
20 REM TRANSPORT DER MATRIX A IN DIE MATRIX B
30 REM
40 REM  A                          B
50 REM 12  13  14          12  15  18  21  24
60 REM 15  16  17          13  16  19  22  25
70 REM 18  19  20  ======> 14  17  20  23  26
80 REM 21  22  23
90 REM 24  25  26
100 REM
110 REM WERTE FUER DIE MATRIX
```

```
120 DATA 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26

130 DIM A(3,5),B(5,3):DEFINT I,J:REM DIM (ZEIL
E,SPALTE)

140 REM EINLESEN A
150 FOR I=3 TO 3:FOR J=1 TO 5:READ A(I,J):NEXT
J,I

160 REM ANZEIGEN A-MATRIX
170 CLS:PRINT"A-MATRIX
180 FOR I=1 TO 3:FOR J=1 TO 5:PRINTUSING"#####
.#";A(I,J);:NEXT J:
PRINT:NEXT I

190 REM TRANSPORT A- NACH B-MATRIX
200 FOR I=1 TO 3:FOR J=1 TO 5:B(J,I)=A(I,J):NE
XT J,I

210 REM ANZEIGE B-MATRIX
220 PRINT"
B-MATRIX
230 FOR I=1 TO 5:FOR J=1 TO 3:PRINTUSING"#####
.#";B(I,J);:NEXT J:
PRINT:NEXT I
240 END
```

2. Fallstudie, Aufgabenstellung

In Ihrer Firma arbeiten 103 Mitarbeiter in verschiedenen Abteilungen. Sie sollen eine Telefonliste erstellen, die für jeden Mitarbeiter folgendes enthält:

- Anrede
- Name
- Abteilung
- Telefonnummer
- Zimmernummer

Die Liste soll mehrfach ausgedruckt werden. Dabei soll sie einmal

nach Namen, einmal nach Abteilungen, einmal nach Telefonnummern und einmal nach Zimmernummern aufsteigend sortiert sein.

Die Sortierfolge ist also:

- Namen, innerhalb des Namens nach Abteilungen
- Abteilung, innerhalb der Abteilung nach Namen
- Zimmernummer, innerhalb der Zimmernummer nach Namen, innerhalb der Namen nach Abteilung
- Telefonnummer, innerhalb der Telefonnummer nach Namen, innerhalb der Namen nach Abteilung

Um insbesondere die Liste nicht viermal schreiben lassen zu müssen, treffen Sie eine Entscheidung:
Wir schreiben die Liste nur einmal und geben sie in unseren Computer ein. Dazu erstellen wir ein kleines Programm, um die Liste in den verschiedenen Sortierfolgen auszudrucken.

Bevor Sie jedoch beginnen, Ihre Idee in die Tat umzusetzen, prüfen Sie ab, ob die Daten überhaupt als DATA-Befehle in den Speicher Ihres Computers passen. Denn DATA-Zeilen haben die unangenehme Eigenschaft, daß Sie zweimal Speicherplatz benötigen:

- In der DATA-Zeile selbst
- In den Feldern und Variablen des Programmes

Ihre Schätzung könnte so aussehen:

1 Zeile pro Name		5 Bytes
Schlüsselwort DATA und Space		2 Bytes
Anrede und Komma		2 Bytes
3 Stellen Telefonnummer und Komma		4 Bytes
3 Stellen Raumnummer und Komma		4 Bytes
14 Stellen Abteilung und Komma		15 Bytes
14 Stellen Name und Komma		15 Bytes
Summe		47 Bytes
103 Mitarbeiter 47*103 =		4841 Bytes
verdoppeln wegen DATA	ca.	10000 Bytes

Die Daten passen also in den Speicher.

Sie haben sich für eine Erfassung mit DATA entschieden, weil Sie sich auf diese Art die Erstellung eines Erfassungsprogrammes sparen können. Sie glauben auch, daß es so einfach geht, denn die Änderungen im Haustelefonverzeichnis sind selten.

Sie stellen zunächst die Plausibilierungsmöglichkeiten zusammen. Denn auch dann, wenn die Werte als DATA im Programm verankert sind, können sie verkehrt sein:

Alle Telefonnummern sind 3-stellig und liegen im Bereich von 200 bis 299 einschliesslich.

Ausnahmen:
Die Telefonzentrale hat die Nr. 7, der Pförtner die Nummer 9. Die beiden Notrufnummern 111 und 112 lassen sich von jedem Telefon aus wählen. Das geht auf einen Verbesserungsvorschlag eines Mitarbeiters zurück, der Bursche hat dafür eine steuerfreie Prämie von DM 75,– kassiert . . . .

Als Raumnummern gibt es:

Erdgeschoss	10 bis 20
1. Stock	100 bis 120
2. Stock	200 bis 220
3. Stock	300 bis 320

Auf Wunsch Ihres Chefs, mit dem Sie das Projekt durchdiskutieren, soll für Raum 20 eine Bezeichnung verwendet werden: EMPFANG.
Sie hoffen, daß niemand auch noch auf die Idee kommt, Raum 112 mit KANTINE zu bezeichnen.

Damit niemand weitere Verbesserungsvorschläge hinsichtlich der Notrufnummern 111 und 112 machen kann, kommt Ihre Frau, mit der Sie das Problem bei einigen Flaschen Trollinger und der Interessanten Fernsehmagazinsendung "PANORAMA" im Hintergrund beklagen, auf eine geniale Idee. Sie greift zum Kugelschreiber und sagt:

"Die Notrufe müssen in Dein Verzeichnis mit Hinein: Etwa so:"

Name	Abteilung	Telefon	Raum
Polizei	Notruf	111	extern
Feuerwehr	Notruf	112	extern
Notfall	Feuerwehr	112	extern
Überfall	Polizei	111	extern
Einbruch	Polizei	111	extern
Überfall	Pförtner	9	Empfang
Unfall	Pförtner	9	Empfang
Unfall	Feuerwehr	112	extern
Feuer	Pförtner	9	Empfang
Feuer	Feuerwehr	112	extern

Nachdem jede Diskussion mit Anderen tolle Ideen gebracht hat, die das Arbeitsergebnis zwar verbessern, jedoch die Programmerstellung immer komplizierter machen, entschliessen Sie sich dazu, die Sache ganz schnell fertigzustellen, bevor irgend jemand anderes noch weitere Ideen einbringen kann.

Bevor diese Lektion zu Ende ist, sei darauf hingewiesen, daß die letzte Entscheidung zwar verständlich ist, jedoch normalerweise verkehrt ist. Man soll Ideen keinesfalls unterdrücken, denn sonst können ständig Änderunganforderungen kommen weil man dieses und jenes nicht berücksichtigt hat. Für diese kleine Applikation mag es so gehen, bei großen Projekten kann eine solche Vorgehensweise der Abkapselung von Ideen zu Zeitverzögerungen oder — im schlimmsten Fall — zur Nichtnutzung des Projektergebnisses führen.

Man sieht auch, daß kleine Sonderwünsche (wie "EMPFANG") einen wesentlichen Einfluß auf das Programm haben könnten: Der Sonderwunsch "EMPFANG" zwingt dazu, die Raumnummer entweder als String zu verarbeiten, oder gezielte IF-Abfragen vorzunehmen. Da aber zu befürchten ist, daß noch andere Abteilungen auf eine solche Idee kommen (z. B. die KANTINE) ist eine Verarbeitung als String besser.

Wie geht es weiter

Das Programm zur 2. Fallstudie ist umfangreich. Wir werden uns die Arbeit erleichtern, indem wir zunächst lernen, wie editiert wird.

6. Lektion

6. LEKTION

Editieren (Erfassen) von Programmen

Befehlsmodus: LIST

Diesen Befehl kennen wir bereits. Er zeigt das sich im Speicher befindliche Programm an.

Beispiele:

LIST 10	zeigt nur Zeile 10
LIST 10–100	zeigt Zeile 10 bis 100
LIST	zeigt das ganze Programm

Das Scrolling des Bildschirmes kann mit Shift @ gestoppt werden. Nach Druck auf die nächste Taste setzt das Scrolling fort.

Befehlsmodus: AUTO
AUTO ermöglicht das automatische Erzeugen von Zeilennummern.

Beispiel:

AUTO	beginnt mit 10, danach 20, 30, 40, . . .
AUTO 100	beginnt mit 100, danach 110, 120, 130, . . .
AUTO 200,100	beginnt mit 200, danach 300, 400, 500, . . .

Ein * neben der generierten Zeilennummer zeigt an, daß die betreffende Zeilennummer schon gespeichert ist. Ein Druck auf die Taste BREAK beendet die Auto-Funktion ohne die bereits vorhandene Zeile zu ändern.

Der Editiermodus:
EDIT Zeilennummer schaltet um vom Befehlsmodus in den Editiermodus. Als Bestätigung wird links die gewünschte Zeilennummer gezeigt.

EDIT 100 eröffnet das Editieren der Zeile 100
EDIT . eröffnet das Editieren der zuletzt angezeigten Zeile

Im Editiermodus dient die Space-Taste als Cursor-Taste nach rechts, die Backspace-Taste dient als Cursor-Taste nach links.

NEW LINE beendet den Editiermodus und stellt die veränderte Zeile in den Speicher zurück.

Die Steuerung erfolgt über Kontrollbuchstaben:

X gehe zum Zeilenende und ermögliche Zusätze. Backspace soll löschen.
H "Hacke" Zeile ab der Cursorposition ab, ermögliche Zusätze, Backspace soll löschen.
I füge ab Cursorpositon ein (Insert).
Q gehe in den Befehlsmodus zurück ohne die Änderungen anzunehmen, denn ich habe "Quatsch" eingetippt . . .

Bei den folgenden Kontrollbuchstaben kann eine Zahl n vorweggehen. Fehlt n, so wird eine 1 angenommen:

nC ändere ab der Cursorposition n Zeichen (Change).
nD lösche ab der Cursorposition n Zeichen (Delete).
nSc suche ab der Cursorposition des n-te Zeichen c (Seek).
nKc lösche ab der Cursorposition die Zeile bis zum n-ten Auftreten des Zeichens c (Kill).

Fortsetzung der 2. Fallstudie:

Damit Sie das Editieren ordentlich üben können, finden Sie einen Teil der Telefonliste nachfolgend als DATA-Befehle abgedruckt.

Ein Tip zum Anfangen: AUTO 2000
Und wenn Sie fertig sind mit dem Erfassen: Speichern Sie die erfassten Daten bitte auf Kassette (oder Diskette), damit Sie diese Erfassung

bei der nächsten Lektion nicht wiederholen müssen! Wie es zu machen ist, steht im Bedienungshandbuch Ihres Video Genie oder TRS-80.

```
2000 REM ANREDE,NAME, ABTEILUNG, TELEFON, ZIMM
ER
2010 DATA ,POLIZEI,NOTRUF,111,EXTERN
2020 DATA ,FEUERWEHR,NOTRUF,112,EXTERN
2030 DATA ,NOTFALL,FEUERWEHR,112,EXTERN
2040 DATA ,NOTFALL,POLIZEI,111,EXTERN
2050 DATA ,UEBERFALL,POLIZEI,111,EXTERN
2060 DATA ,UEBERFALL,PFOERTNER,9,EMPFANG
2070 DATA ,UNFALL,PFOERTNER,9,EMPFANG
2080 DATA ,FEUER,PFOERNER,9,EMPFANG
2090 DATA ,FEUER,FEUERWEHR,112,EXTERN
2100 DATA H,LUCE,ORG-DV,232,15
2110 DATA F,LUCE,KUECHE,250,120
2120 DATA H,HENSEL,EINKAUF,201,311
2130 DATA H,VERMISSEN,ADM-LEITUNG,200,302
2140 DATA F,STROHSACKER,SEKR.ADM-LEITUNG,202,3
03
2150 DATA F,PRANTNER,PFOERTNER,9,EMPFANG
2160 DATA H,KLANG,PFOERTNER,9,EMPFANG
2170 DATA H,MUELLER,PFOERTNER,9,EMPFANG
2180 DATA F,MEIER,ZENTRALE,7,10
2190 DATA H,NIRSCHL,KANTINE,110,222
2200 DATA F,SCHMID,SCHREIBDIENST,201,208
2210 DATA F,HILLER,SCHREIBDIENST,201,208
2220 DATA F,SORG,SCHREIBDIENST,201,209
2230 DATA F,BUERS,SCHREIBDIENST,201,270
2240 DATA F,PROELLER,SCHREIBDIENST,201,208
2250 DATA H,VISIKO,PERSONALABT,250,305
2260 DATA F,KNUELL,PERSONALABT,250,305
2270 DATA H,KOSS,KANTINE,206,118
2280 DATA H,VETTER,EINKAUF,208,119
2290 DATA H,BOLLIG,BUCHALTUNG,305,222
2300 DATA F,MALZAHN,BUCHALTUNG,305,221
2310 DATA H,KRETSCHMER,VERKAUFSADM,203,270
2320 DATA H,SCHLEBUSCH,VERKAUFSADM,204,272
2330 DATA F,APFELSIEPER,VERKAUFSADM,204,272
2340 DATA H,SCHWABL,VERKAUFSADM,204,271
2350 DATA FRL,BUCHERT,LEHRLING,299,12
2360 DATA H,THEIN,LEHRLING,299,12
```

```
2370 DATA FRL,REIMANN,LEHRLING,299,12
2380 DATA H,HERZOG,HAUSMEISTER,200,15
2390 DATA F,HERZOG,HAUSMEISTER,200,15
2400 DATA H,MENGEL,SCHLOSSEREI,224,17
2410 DATA FRL,KLINGSAM,ORG-DV,280,302
2420 DATA H,DENZ,ORG-DV,282,302
2430 DATA F,SPITZMIR,TECHN-ABT,275,301
2440 DATA H,BLEIKRANTZ,TECHN-ABT,275,301
2450 DATA FRL,MUERRKNALL,TECHN-ABT,275,300
2460 DATA H,MEYER,BUEROORG,276,303
2470 DATA F,LINDE,BUEROORG,277,303
4000 DATA 9999:REM KENNZEICHNET DATENENDE
```

Die Daten enthalten einige Fehler. Da unser Programm jedoch eine Plausibilierung der Eingabedaten enthalten wird, lassen wir die Fehler in den Daten und vertrauen darauf, daß das Programm uns die Fehler meldet.

Vor Fortsetzung der 2. Fallstudie müssen wir uns mit Strings beschäftigen. Angewendet haben wir Strings schon mehrmals, aber es fehlt noch die Theorie dazu.

7. Lektion

7. LEKTION

Strings

Das englische Wort "String" wird in BASIC-Hand- und -Fachbüchern gerne mit "Zeichenkette" übersetzt. Diese Übersetzung ist nicht korrekt, denn die Abspeicherung einer Zeichenkette ist nur eine mögliche Nutzung von Strings. Strings bieten die Möglichkeit, jede beliebige Bitkombination (hexadezimal 00 bis FF) zu verarbeiten.

Stringvariablen erkennen wir an dem Suffix $.

Werte für Strings stehen häufig in "-Paaren. Wir haben jedoch bei DATA gesehen, daß Werte für Strings auch ohne " zugewiesen werden können.

Beim BASIC-Dialekt des Video-Genie oder TRS-80 kann ein String zwischen 0 bis 255 Bytes lang sein. Hat ein String die Länge 0, so nennt man ihn auch "Leerstring".

Die Länge eines Strings kann man wie folgt feststellen:

Übung (im Befehlsmodus):

A$="12345":?A$:?LEN(A$)

Anzeige:

12345	Inhalt des Strings
5	Länge des Strings

Mit Strings kann nicht gerechnet werden – kein Wunder, denn ein String kann ja alles mögliche außer Ziffern enthalten.

Strings können verkettet werden. Hierzu wird das plus (+)-Zeichen "mißbraucht":

Übung (im Befehlsmodus):

```
A$="12345":B$="ABCDE":C$=A$+"."+B$:?C$:?LEN(C$)
```

Anzeige:

12345.ABCDE	Inhalt des Strings C$
11	Länge von C$

Strings können miteinander verglichen werden. Ein Vergleich eines Strings mit einem numerischen Wert ist nicht zulässig.

Übung (im Befehlsmodus):

A$="ABCDE":B$="ABCDE":?A$=B$	Anzeige: –1 für wahr
C$=" ABCDE":?A$=C$	Anzeige: 0 für unwahr
A=12345:?A=A$	Anzeige: eine Fehlermeldung

Zwei Strings sind nur dann gleich, wenn Sie den gleichen Inhalt und die gleiche Länge haben.

Auf Teile eines Strings kann man mit MID$ zugreifen. Die allgemeine Schreibweise ist:

MID$(Stringname, ab Position, Anzahl Bytes)

Die erste Position eines Strings hat die Positonsnummer 1.

Übung (im Direktmodus):

```
A$="12345":FOR I=1 TO LEN(A$):?MID$(A$,I,1):NE
XT
```

Anzeige:

1
2
3
4
5

Für Zugriffe auf den rechten oder linken Rand eines Strings gibt es die Sonderbefehle LEFT$ und RIGHT$. So ist

LEFT$(A$,4) funktionsgleich mit MID$(A$,1,4) und
RIGHT$(A$,2) funktionsgleich mit MID$(A$,LEN(A$)−1,2).

Nun, dies wären bereits die wesentlichen Grundlagen der Stringverarbeitung mit dem im Video Genie und TRS-80 verwendeten BASIC-Dialekt.

Es gibt noch einen abweichenden BASIC-Dialekt, bei dem die Stringverarbeitung ganz anders organisiert ist. Bei BASIC-Dialekten, deren Ursprung auf das amerikanische Softwarehaus "Microsoft" zurück geht, erfolgt die Stringverarbeitung jedoch so wie hier geschildert.

Das Programm, welches die erste Realisierungsstufe der 2. Fallstudie darstellt, verwendet einen Befehl zur Wandlung eines Strings in eine numerische Variable in seiner Zeile 72 (siehe Programmlisting auf Seite 50):

VAL

VAL ergibt den Wert der ersten Zahl, die in einer Zeichenkette enthalten ist. Die Zahl – ggf. mit Vorzeichen + oder – – muß linksbündig im String stehen oder über MID$ linksbündig dem VAL-Befehl zugänglich gemacht werden.

Übung (im Befehlsmodus):

?VAL("12")	12
?VAL("+13")	13
?VAL("−14")	−14

?VAL("GROESSE: 1.40M")	0
?VAL("1.40M GROSS")	1.4
?VAL("%")	Fehlernachricht, der BASIC-Interpreter des TRS-80 oder Video Genie kann einen String der mit dem %-Zeichen beginnt nicht mit VAL verarbeiten !!!

Im Ausführungsmodus:

```
1 CLEAR 100:A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ
EIN LANGER SATZ, 1.40M LANG"
2 FOR I=1 TO LEN(A$):J=VAL(MID$(A$,I,LEN(A$)-1
)):IF J<>0 THEN PRINT J:I=LEN(A$)+1

3 NEXT
4 END
```

Wenn Sie dies kurze Programm laufen lassen (es sucht die Zahl), so achten Sie darauf, ob Sie die Laufzeit bemerken.

Ergebnis:
Die Laufzeit ist deutlich spürbar (1 bis 2 Sekunden).

Generelles zur Laufzeit

Stringoperationen sind etwa so langsam wie mathematische Operationen. Deshalb sollte man bemüht sein, komplexe Stringoperationen in FOR-TO-Schleifen zu vermeiden.

Die folgende Tabelle enthält einige Laufzeitangaben. Beachten Sie jedoch, daß Laufzeitangaben bei BASIC nur eine Tendenz angeben können: die Laufzeit eines Programmes wird wesentlich vom Zugriff auf den Variablenpool bestimmt. Ein Variablenpool läßt sich bei der Erstellung von Tabellen wie der folgenden nicht berücksichtigen:

Befehl	Laufzeit	Bemerkung
A=B	3.2 ms	
A=1	3.3 ms	
C=A*B	7 ms	A=10, B=200
C=10*200	8 ms	
C=200*10	7 ms	
C=A%*B%	8 ms	A%=10, B%=200
A=VAL("1")	6 ms	
A=VAL("32000")	18 ms	
A=VAL(A$)	18 ms	Wert von A$="32000"
A$=MID$(B$,2,3)	8 ms	Wert von B$="12345"
A$=A$+B$:A$=LEFT$(A$,4)	13 ms	Wert von A$="1234", B$="ABCDE"
A=LOG(300)	26 ms	
je Space	0.02 ms	Space nicht in REMs
je Zeilenwechsel	0.05 ms	

Fortsetzung der 2. Fallstudie:

In der ersten Entwicklungsstufe des Programmes sind etliche Stringoperationen enthalten. Die DATA-Statements ab Zeile 2000 entnehmen Sie bitte der Editierübung von der 6. Lektion: sie gehören mit zum Programm. Da Sie bei der 6. Lektion die Daten auf Kassette aufgezeichnet haben, müssen Sie diese Daten erst in den Computer zurückladen bevor Sie mit der Erfassung der folgenden Programmzeilen beginnen:

```
1 CLEAR 5000:REM PLATZ FUER DIE STRINGS
10 CLS:PRINT"TELEFONLISTE, SORTIERT NACH VERSC
HIEDENEN BEGRIFFEN
15 DEFINT A-Z
20 M=140:DIM A$(M),N$(M),ABT$(M),T(M),ZI$(M):R
EM M=ANZAHL MITARBEITER
30 REM A$(M)=ANREDE, N$(M)=NAME, ABT$(M)=ABTEI
LUNG, T(M)=TELEFONNR.,
 ZI$(M)=ZIMMERNR.

35 REM ERSTES EINLESEN DABEI PLAUSI DER DATEN.
```

```
40 PRINT"EINLESEN UND PLAUSIBILIEREN":FOR I=1
TO M:PRINT@960,I;:
READ A$(I):IF A$(I)="9999"THEN I=M+1:GOTO 80:R
EM DATENENDE
50 IF A$(I)<>"" AND A$(I)<>"F"AND A$(I)<>"H" A
ND A$(I)<>"FRL"
THEN PRINTA$;"ANREDE ("2000+10*I;") UNZULAESSI
G":INPUT A$
55 READ N$(I),ABT$(I),T(I),ZI$(I)
60 T=T(I):IF T<200 OR T>299 THEN IF NOT T=7 AN
D NOT T=9 AND NOT T=111
AND NOT T=112 PRINTT(I);" TELEFONNR. ("2000+10*
I;") Unzulaessig.":INPUT A$
70 ZI$=ZI$(I):IF ZI$="EMPFANG" OR ZI$="EXTERN"
THEN 80
72 ZI=VAL(ZI$):IF ZI>=10 AND ZI<=20
OR ZI>=100 AND ZI<=120
OR ZI>=200 AND ZI<=220
OR ZI>=300 AND ZI<=320 THEN 80
75 PRINTZI$;" ZIMMERNR. ("2000+I*10;") UNZULAE
SSIG":INPUT A$
80 NEXT

999 REM AUSGABE DER LISTE
1000 RESTORE:PRINT"TELEFONVERZEICHNIS DER KLIN
GELDREHER A.G., STAND: ";D$
1010 PRINT:PRINT"NAME","ABTEILUNG","TELEFON","
ZIMMER"
1020 FOR I=1 TO M:READ A$:IF A$="9999" THEN PR
INT:PRINT:PRINT:RETURN
1025 IF A$="" THEN A$="     " ELSE A$=LEFT$(A$+
"."+"    ",4)
1030 READ N$,ABT$,T,ZI$
1032 IF LEN(N$)>15 THEN N$=LEFT$(N$,15))
1034 IF LEN(ABT$)>15 THEN ABT$=LEFT$(ABT$,15)
1040 PRINTA$;N$,ABT$,T,ZI$:NEXT
```

Der oben abgedruckte Programmabschnitt plausibiert die Daten. Der Programmablauf endet mit einem Fehlerhinweis (Return ohne Gosub), der sich durch zusätzliches Einfügen der Zeilen

```
90 GOSUB 1000
100 STOP
```

entfernen läßt.

Der RETURN-Ausgang in Zeile 1020 ist eigentlich nicht zulässig, vergl. dazu die Abhandlung über die FOR-TO-Problematik (4. Lektion). PRINT @ in Zeile 40 bewirkt die Anzeige eines "digitalen Zählwerkes" in der 960. Bildschirmposition (960/64=15 Rest 0, d. h., die Anzeige erfolgt am Anfang (Rest 0) der 15. Bildschirmzeile).

Bevor Sie den Computer abschalten, speichern Sie bitte das Programm erneut auf Kassette (oder Diskette), denn es wird bei der nächsten Lektion noch benötigt.

NOTIZEN

8. Lektion

8. LEKTION

Im Rahmen unserer 2. Fallstudie werden wir zuerst die am Ende der letzten Lektion erarbeitete Teillösung diskutieren:

Zeile 10: Zeigt dem Bediener, welches Programm er gestartet hat.

Zeile 20: Deklariert die Felder. Diese Zeile hat auch Dokumentationscharacter.

Zeile 40: Auch wenn die Daten als DATA-Statements Teile des Programmes sind, können sie fehlerhaft sein und müssen geprüft werden. Das Einlesen und Prüfen der Daten dauert mehrere Sekunden. Damit der Bediener erfährt, was der Computer so lange macht, wird ihm ein Schirmbild gezeigt, das im oberen Teil der Tätigkeit (EINLESEN UND PLAUSIBILIEREN) angibt. Im unteren Teil – genau auf der 960. Bildschirmstelle beginnend wird die laufende Nummer der gerade bearbeiteten Zeile gezeigt. (Der Bildschirm hat 16 Zeilen zu je 64 Stellen. Die 960. Stelle ist genau die linke Position der 16. Zeile. Das Semikolon verhindert einen Zeilenvorschub, so daß ein stehendes Bild mit einer Anzeige des sich in Verarbeitung befindlichen Satzes im Stil einer Digitalanzeige entsteht.)
Unmittelbar mit dem Einlesen des ersten Feldes A$ ist die Abfrage auf Datenende verbunden.
Zeile 50 prüft die Anrede. Als Anrede ist auch ein Leerstring zugelassen.

Zeile 55 liest die restlichen Daten des Satzes ein. Bei gleichem Index bilden somit A$(I),N$(I),ABT$(I),T(I) und ZI$(I) die zusammengehörigen Variablen eines "Datensatzes".

Das Programm enthält beim Einlesen T(I) in Zeile 50 einen Programmfehler. Dieser Fehler ist aus Unterrichtsgründen eingebaut, damit später ein Unterschied beim Sortieren demonstriert werden kann. Um diesen Fehler zu demonstrieren, ändern wir in Zeile 2010 die Telefonnummer in A111 und lassen das Programm laufen:

Ergebnis:
Dieser Fehlerhinweis ist einem Bediener kaum zuzumuten !
(Bitte ändern Sie Zeile 2010 wieder zurück.)

Zeile 60 prüft die Telefonnummer. Ist sie unzuläßig, wird die Zeilennummer des DATA-Statements mit einem Fehlerhinweis gezeigt. Der Bediener quittiert durch Druck auf die NEW-LINE-Taste. Die Zuweisung T=T(I) am Anfang der Zeile erfolgt, damit bei den einzelenen Abfragen T geschrieben werden kann, und nicht T(I) geschrieben werden muß.

Zeile 70 prüft die beiden möglichen "Zimmernummern" die keine Nummern sind.

Zeile 72 prüft die zulässigen Nummernkreise der Zimmernummer. Durch entsprechende Schreibweise wird trotz der verschachtelten IF-Abfrage eine gewisse Übersichtlichkeit erreicht. Der String wird mit ZI=VAL(ZI$(I)) (jedenfalls von der Wirkung her) einer numerischen Variablen zugewiesen.

Zeile 75 ist die Fehleranzeige im Falle einer nicht zulässigen Zimmernummer.

Ab Zeile 1000 beginnt ein Unterprogramm, das die Telefonliste auf dem Bildschirm in der Reihenfolge zeigt, wie sie bei DATA ab Zeile 2000 festgelegt ist. In D$ soll später das Tagesdatum stehen.

Zeile 1025 macht die Anrede einheitlich 4 Bytes lang. Sofern die zugelassenen Anreden F, FRL oder H enthalten sind, wird hier daraus gemacht werden: F., FRL. oder H. – mit 2 Spaces hinter H. und F. und keinem Space hinter FRL. .

Zeile 1032 und 1034 verkürzt längere Namen und Abteilungsbe-

zeichnungen auf maximal 15 Stellen.

Zeile 1040 führt die Anzeige auf dem Bildschirm durch. Dabei werden die Tabellierstellen mit Hilfe der Kommata genutzt.

Bubblesort

Die nächste Realisierungsstufe unseres Fallstudienprogrammes führt bereits die beiden ersten gewünschten Sortierfolgen nach Namen und nach Abteilung als Hauptsortierbegriff durch. Das angewendete Sortierverfahren hat den Namen "Bubblesort". Es ist von Dr. phil. N. Wirth ("PASCAL") in seinem Buch "Algorithmen und Datenstrukturen, 1975" so beschrieben worden (Array ist der englische Ausdruck für Feld):

"Sortieren durch direktes Austauschen.

. . . legen wir eine Methode dar, in der Austausch von zwei Elementen die wesentliche Charakteristik des Prozesses ist. Der . . . Algorithmus . . . beruht auf dem Prinzip des fortgesetzten Vergleichens und Austauschens von Paaren nebeneinanderliegender Elemente, bis alle Elemente sortiert sind.
. . . durchlaufen wir das Array mehrmals und lassen jedesmal das kleinste Element der restlichen Menge zum linken Ende des Arrays wandern. Wenn wir zur Abwechslung den Array vertikal statt horizontal betrachten und – mit einiger Phantasie – die Elemente als Blasen (bubbles) auffassen, so steigt bei jedem Durchlauf durch das Array eine Blase auf die ihrem Gewicht entsprechende Stufe."
. . . Offensichtlich kann . . . verbessert werden, wenn man sich merkt, ob während eines Durchlaufes überhaupt ein Austausch stattgefunden hat."

Genau diesen Bubblesort wenden wir an:

```
1 CLEAR 5000:REM PLATZ FUER DIE STRINGS
10 CLS:PRINT"TELEFONLISTE, SORTIERT NACH VERSC
HIEDENEN BEGRIFFEN
20 M=140:DIM A$(M),N$(M),ABT$(M),T(M),ZI$(M):R
EM M=ANZAHL MITARBEITER
30 REM A$(M)=ANREDE, N$(M)=NAME, ABT$(M)=ABTEI
LUNG, T(M)=TELEFONNR.,
 ZI$(M)=ZIMMERNR.

35 REM ERSTES EINLESEN DABEI PLAUSI DER DATEN.
```

```
40 PRINT"EINLESEN UND PLAUSIBILIEREN":FOR I=1
TO M:PRINT@960,I;:
READ A$(I):IF A$(I)="9999"THEN I=M+1:GOTO 80:R
EM DATENENDE
50 IF A$(I)<>"" AND A$(I)<>"F"AND A$(I)<>"H" A
ND A$(I)<>"FRL"
THEN PRINTA$;"ANREDE ("2000+10*I;") UNZULAESSI
G":INPUT A$
55 READ N$(I),ABT$(I),T(I),ZI$(I)
60 T=T(I):IF T<200 OR T>299 THEN IF NOT T=7 AN
D NOT T=9 AND NOT T=111
AND NOT T=112 PRINTT(I);" TELEFONNR. ("2000+10*
I;") Unzulaessig.":INPUT A$
70 ZI$=ZI$(I):IF ZI$="EMPFANG" OR ZI$="EXTERN"
THEN 80
72 ZI=VAL(ZI$):IF ZI>=10 AND ZI<=20
OR ZI>=100 AND ZI<=120
OR ZI>=200 AND ZI<=220
OR ZI>=300 AND ZI<=320 THEN 80
75 PRINTZI$;" ZIMMERNR. ("2000+I*10;") UNZULAE
SSIG":INPUT A$
80 NEXT

100 REM BUBBLESORT. J=ANZAHL DURCHGAENGE, S=ME
RKER FUER SORTIERT
105 REM 1.SORT: NAME, ABTEILUNG
110 CLS:PRINT"SORT NACH NAMEN":FOR I=1 TO Z-2:
PRINT@960,J,I;:
IF N$(I)<N$(I+1) THEN 150:REM KEIN TAUSCH NOE
TIG
120 IF N$(I)>N$(I+1) THEN GOSUB 1200 ELSE IF A
BT$(I)>ABT$(I+1) GOSUB 1200
150 NEXT
160 J=J+1:IF S=1 THEN S=0:GOTO 110:REM WENN NI
CHT SORTIERT

170 GOSUB 1000:REM AUSDRUCK

200 REM 2.SORT: ABTEILUNG, NAME
205 J=0:REM DURCHGANGSZAEHLER ZURUECK
210 CLS:PRINT"SORT NACH ABT.":FOR I=1 TO Z-2:P
RINT@960,J,I;:
```

```
IF ABT$(I)<ABT$(I+1) THEN 230:REM KEIN TAUSCH
NOETIG
220 IF ABT$(I)>ABT$(I+1) THEN GOSUB 1200
ELSE IF N$(I)>N$(I+1) GOSUB 1200
230 NEXT
240 J=J+1:IF S=1 THEN S=0:GOTO 210:REM WENN NI
CHT SORTIERT

270 GOSUB 1000:REM AUSDRUCK
280 STOP

999 REM AUSGABE DER LISTE
1000 RESTORE:PRINT"TELEFONVERZEICHNIS DER KLIN
GELDREHER A.G., STAND: ";D$
1010 PRINT:PRINT"NAME","ABTEILUNG","TELEFON","
ZIMMER"
1020 FOR I=1 TO Z-1:IF A$(I)="9999" THEN PRINT
:PRINT:PRINT:RETURN
1025 IF A$(I)="" THEN A$="     " ELSE A$=LEFT$(
A$(I)+","+"    ",4)
1030 N$=N$(I):ABT$=ABT$(I):T=T(I):ZI$=ZI$(I)
1032 IF LEN(N$)>15 THEN N$=LEFT$(N$,15))
1034 IF LEN(ABT$)>15 THEN ABT$=LEFT$(ABT$,15)
1040 PRINTA$;N$,ABT$,T,ZI$:NEXT

1199 REM BUBBLESORT, VERTAUSCHEN MIT ZWISCHENB
EREICH
1200 S=1:A$=A$(I):N$=N$(I):ABT$=ABT$(I):T=T(I)
:ZI$=ZI$(I)
1210 I1=I+1:A$(I)=A$(I1):N$(I)=N$(I1):ABT$(I)=
ABT$(I1):T(I)=T(I1):ZI$(I)=ZI$(I1)
1220 A$(I1)=A$:N$(I1)=N$:ABT$(I1)=ABT$:T(I1)=T:
ZI$(I1)=ZI$:RETURN

2000 REM
```

ab hier DATA-Zeilen, wie bereits bekannt . . . .

Diskussion des Programmes:
Zeile 15: Da das Programm keinerlei Rechnungen durchführt, können alle numerischen Variablen als ganzzahlig deklariert werden. Durch diese Maßnahme wird Speicherplatz und Laufzeit gespart. Zeile 110 enthält am Ende den Vergleich zweier Elemente des Feldes. Ist das linke

Element kleiner, so ist kein Austausch der beiden Elemente erforderlich.

Zeile 120: Ist dagegen das linke Element größer, so ist ein Austausch nötig. Hinter ELSE wird der Fall verarbeitet, daß der primäre Sortierbegriff (Name) gleich ist: Dann wird der Sortierbegriff um den sekundären Sortierbegriff (Abteilung) erweitert.

Zeile 160: Am Wert von S erkennen wir, ob vertauscht wurde. Wurde nicht vertauscht, so ist das Feld vollständig sortiert. Andernfalls erfolgt ein weiterer Durchlauf. Die Variable "S" enthält nur zwei Zustände (Ja oder Nein). Sie wird deshalb auch "Schalter" genannt.

Zeile 1200 ff: Hier erfolgt der Austausch der Elemente. Die Codierung erfolgt als Unterprogramm, da insgesamt nach 4 Kriterien sortiert werden soll und jede Sortierung wird dieses Unterprogramm nutzen können.

Zeile 1000 ff: Das Ausgabeunterprogramm wurde geringfügig verändert. Es zeigt jetzt das sortierte Feld, nicht mehr die Daten in der Reihenfolge wie in DATA.

Übung:
Bitte, ermitteln Sie die Laufzeit der Programmabschnitte ohne die Ausgabezeit der Telefonliste.

Abschnitt	Laufzeit
Plausibilierung	8 s
Sort nach Namen	157 s
Sort nach Abteilung	156 s
gesamt	321 s

Sortieren durch direktes Einfügen

Zu siesem Sortierverfahren schreibt Dr. phil. N. Wirth: "Diese Methode wird von Kartenspielern mit Vorliebe benutzt. Die Elemente (Karten) werden begrifflich in eine Ziel-Sequenz . . . und eine Quellensequenz . . . aufgeteilt. . . . wird mit jedem Schritt das . . . Element der Quellen-

sequenz herausgegriffen und in die Zielsequenz an die entsprechende Stelle eingefügt.''

Dieses Verfahren wenden wir in unerer zweiten Teillösung der 2. Fallstudie an: Wir wollen einige Sortierverfahren miteinander vergleichen und für die fertige Lösung die uns am günstigsten erscheinende Sortiermethode anwenden.

Das geänderte Programm mit Sortieren durch direktes Einfügen ist nachfolgend abgedruckt:

```
1 CLEAR 5000:REM PLATZ FUER DIE STRINGS
10 CLS:PRINT"TELEFONLISTE, SORTIERT NACH VERSC
HIEDENEN BEGRIFFEN
15 DEFINT A-Z
20 M=140:DIM A$(M),N$(M),ABT$(M),T(M),ZI$(M):R
EM M=ANZAHL MITARBEITER
30 REM A$(M)=ANREDE, N$(M)=NAME, ABT$(M)=ABTEI
LUNG, T(M)=TELEFONNR.,
 ZI$(M)=ZIMMERNR.

35 REM ERSTES EINLESEN. DABEI PLAUSI DER DATEN
, EINSTELLEN IN TABELLE SORTIERT NACH NAMEN
40 PRINT"PLAUSIBILIEREN, EINSTELLEN NACH NAMEN
:FOR I=1 TO M:PRINT@960,I;:
READ A$(I):IF A$(I)="9999"THEN I=M+1:GOTO 95:R
EM DATENENDE
50 IF A$(I)<>"" AND A$(I)<>"F"AND A$(I)<>"H" A
ND A$(I)<>"FRL"
THEN PRINTA$;"ANREDE ("2000+10*I;") UNZULAESSI
G":INPUT A$
55 READ N$(I),ABT$(I),T(I),ZI$(I)
60 T=T(I):IF T<200 OR T>299 THEN IF NOT T=7 AN
D NOT T=9 AND NOT T=111
AND NOT T=112 PRINTT(I);" TELEFONNR. ("2000+10*
I;") Unzulaessig.":INPUT A$
70 ZI$=ZI$(I):IF ZI$="EMPFANG" OR ZI$="EXTERN"
THEN 80
72 ZI=VAL(ZI$):IF ZI>=10 AND ZI<=20
OR ZI>=100 AND ZI<=120
OR ZI>=200 AND ZI<=220
OR ZI>=300 AND ZI<=320 THEN 80
```

```
75 PRINTZI$;" ZIMMERNR. ("2000+I*10;") UNZULAE
SSIG":INPUT A$
80 IF I=1 THEN 95:REM KEIN AUFSCHIEBEN WENN NU
R 1 ELEMENT
82 FOR J=1 TO I-1:IF N$(J)<N$(I)THEN 90
84 IF N$(J)>N$(I) GOSUB 1400:J=I:GOTO 90
86 IF N$(J)=N$(I) THEN IF ABT$(J)>ABT$(I) GOSU
B 1400:J=I
90 NEXT J
95 NEXT I

170 GOSUB 1000:REM AUSDRUCK

200 REM 2.EINLESEN: ABTEILUNG, NAME
210 CLS:RESTORE:PRINT"EINLESEN NACH ABT.":FOR
I=1 TO Z-1:PRINT@960,I;
215 READ A$(I),N$(I),ABT$(I),T(I),ZI$(I)
220 IF I=1 THEN 280:REM KEIN AUFSCHIEBEN WENN
NUR 1 ELEMENT
230 FOR J=1 TO I-1:IF ABT$(J)<ABT$(I) THEN 270
240 IF ABT$(J)>ABT$(I) GOSUB 1400:J=I:GOTO 270
250 IF ABT$(J)=ABT$(I) THEN IF N$(J)>N$(I) GOS
UB 1400:J=I
270 NEXT J
280 NEXT I:GOSUB 1000:REM AUSDRUCK
300 STOP
999 REM AUSGABE DER LISTE

999 REM AUSGABE DER LISTE
1000 RESTORE:PRINT"TELEFONVERZEICHNIS DER KLIN
GELDREHER A.G., STAND: ";D$
1010 PRINT:PRINT"NAME","ABTEILUNG","TELEFON","
ZIMMER"
1020 FOR I=1 TO M:READ A$(I):IF A$(I)="9999"
THEN PRINT:PRINT:PRINT:RETURN
1025 IF A$(I)="" THEN A$="     " ELSE A$=LEFT$(
A$(I)+","+"    ",4)
1030 N$=N$(I):ABT$=ABT$(I):T=T(I):ZI$=ZI$(I)
1032 IF LEN(N$)>15 THEN N$=LEFT$(N$,15))
1034 IF LEN(ABT$)>15 THEN ABT$=LEFT$(ABT$,15)
1040 PRINTA$;N$,ABT$,T,ZI$:NEXT

1399 REM EINFUEGEN IN  TABELLENPOSITION
```

```
1400 FOR K=I TO J STEP -1
1410 K1=K+1:A$(K1)=A$(K):N$(K1)=N$(K):ABT$(K1)
=ABT$(K):T(K1)=T(K):ZI$(K1)=ZI$(K):NEXT K

1420 I1=I+1:A$(J)=A$(I1):N$(J)=N$(I1):ABT$(J)
=ABT$(I1):T(J)=T(I1):
ZI$(J)=ZI$(I1):RETURN
```

Diskussion des Programmes:

Die DATA-Zeilen stellen die Quellen-Sequenz dar. Das erste Einfügen nach Name und Abteilung erfolgt bereits bei der Plausibilierung ab Zeile 80.

Zeile 80: Beim ersten Element ist eine Umsorierung in der Ziel-Sequenz nicht erforderlich.

Zeile 82: Das neue Element wird zunächst provisorisch ans rechte Ende der Ziel-Sequenz angefügt. Es wird in der hier beginnenden FOR-TO-Schleife gegen die vorher eingefügten Elemente verglichen bis die Einfügestelle (Zeile 84 oder 86) gefunden ist. Sofern keine Einfügestelle gefunden wurde, steht das Element bereits an der richtigen Stelle, da es der größte bisher aufgetretene Sortierbegriff überhaupt ist.

Zeile 90 und 95: Hier kann man nicht schreiben: NEXT J,I , denn es wird von Zeilen 40 und 80 aus auf Zeile 95 zugesprungen.

Zeilen 1400 ff: Das Einfügen kann erst erfolgen, nachdem eine Lücke in der Ziel-Sequenz erzeugt wurde. Dazu werden in Zeile 1410 alle Elemente einschliesslich des hinzukommenden Elementes von der Einfügestelle ab um eine Position nach rechts verschoben. Die Laufvariable muß von rechts nach links laufen ! Hätte man sie von links nach rechts laufen gelassen, so hätten alle Elemente rechts der Einfügestelle den gleichen Inhalt bekommen !

Zeile 1420: Hier erfolgt der eigentliche Einfügvorgang.

Übung:
Bestimmen Sie die Laufzeiten des Programmes in der Sortierversion mit direktem Einfügen.

Plausibilierung und Sort nach Namen:	52 s
Sort nach Abteilung	43 s
gesamt	95 s

N. Wirth wertet "Das Sortieren durch direktes Einfügen scheint keine sehr geeignete Methode für digitale Rechenanlagen zu sein: Einfügen eines Elementes durch das Verschieben einer ganzen Zeile um eine einzige Position ist eher unwirtschaftlich."

Wir bleiben bei N. Wirth und lassen ihn fortsetzen:

Sortieren durch direktes Auswählen

Diese Methode beruht auf folgenden Prinzipien:

1) Auswahl des Elementes mit dem kleinsten Schlüssel
2) Austausch gegen das erste Element
   Darauf wiederholt man diese Operation mit den restlichen n-1 Elementen, dann mit n-2 Elementen, . . .

Programmlisting:

```
1 CLEAR 5000
10 CLS:PRINT"TELEFONLISTE, SORTIERT NACH VERSC
HIEDENEN BEGRIFFEN
15 DEFINT A-Z
20 T=140:DIM A$(T),N$(T),ABT$(T),T(T),ZI$(T):R
EM T=ANZAHL MITARBEITER
30 REM A$(T)=ANREDE, N$(T)=NAME, ABT$(T)=ABTEI
LUNG, T(T)=TELEFONNR.,
ZI$(T)=ZIMMERNR.

35 REM ERSTES EINLESEN DABEI PLAUSI DER DATEN.
40 PRINT"EINLESEN UND PLAUSIBILIEREN":J=T:FOR
I=1 TO J:PRINT@960,I;:
READ A$(I):IF A$(I)="9999"THEN Z=I-1:I=J+1:GO
TO 80:REM DATENENDE
50 IF A$(I)<>"" AND A$(I)<>"F"AND A$(I)<>"H" A
ND A$(I)<>"FRL"
THEN PRINTA$;"ANREDE ("2000+10*I;") UNZULAESSI
G":INPUT A$
```

```
55 READ N$(I),ABT$(I),T(I),ZI$(I)
60 T=T(I):IF T<200 OR T>299 THEN IF NOT T=7 A
ND NOT T=9 AND NOT T=111
AND NOT T=112 PRINTT(I);" TELEFONNR. ("2000+10*
I;") Unzulaessig.":INPUT A$
70 ZI$=ZI$(I):IF ZI$="EMPFANG" OR ZI$="EXTERN"
THEN 80
72 T=VAL(ZI$):IF T>=10 AND T<=20
OR T>=100 AND T<=120
OR T>=200 AND T<=220
OR T>=300 AND T<=320 THEN 80
75 PRINTZI$;" ZIMMERNR. ("2000+I*10;") UNZULAE
SSIG":INPUT A$
80 NEXT

100 REM 1.SORT: NAME, ABTEILUNG. N$ U. ABT$= Z
WISCHENSPEICHER KLEINSTE ELEMENTE
110 CLS:PRINT"SORT NACH NAMEN":FOR I=1 TO Z-1:
PRINT@960,I;:N$="ZZZZZ":ABT$=N$
120 FOR J=I TO Z:IF N$(J)<N$ THEN N$=N$(J):ABT
$=ABT$(J):T=J:GOTO 130
125 IF N$(J)=N$ THEN IF ABT$(J)<ABT$ N$=N$(J)
:ABT$=ABT$(J):T=J
130 NEXT J:GOSUB 1200:NEXT I
170 GOSUB 1000:REM AUSDRUCK

200 REM 2.SORT: ABTEILUNG, NAME. ABT$ U. N$= Z
WISCHENSPEICHER KLEINSTE ELEMENTE
210 CLS:PRINT"SORT NACH ABT.":FOR I=1 TO Z-1:P
RINT @960,I:N$="ZZZZZ":ABT$=N$
220 FOR J=I TO Z:IF ABT$(J)<ABT$ THEN N$=N$(J)
:ABT$=ABT$(J):T=J:GOTO 230
225 IF ABT$(J)=ABT$ THEN IF ABT$(J)<ABT$ N$=N$
(J):ABT$=ABT$(J):T=J
230 NEXT J:GOSUB 1200:NEXT I
270 GOSUB 1000:REM AUSDRUCK
280 STOP

999 REM AUSGABE DER LISTE
1000 PRINT"TELEFONVERZEICHNIS DER KLINGELDREHE
R A.G., STAND: ";D$
1010 PRINT:PRINT"NAME","ABTEILUNG","TELEFON",
"ZIMMER"
```

```
1020 FOR I=1 TO Z
1025 IF A$(I)="" THEN A$="     " ELSE A$=LEFT$(
A$(I)+"."+"    ",4)
1030 N$=N$(I):ABT$=ABT$(I):T=T(I):ZI$=ZI$(I)
1032 IF LEN(N$)>15 THEN N$=LEFT$(N$,15))
1034 IF LEN(ABT$)>15 THEN ABT$=LEFT$(ABT$,15)
1040 PRINTA$;N$,ABT$,T,ZI$:NEXT:PRINT"

":RETURN

1199 REM  VERTAUSCHEN MIT ZWISCHENBEREICH
1200 A$=A$(I):N$=N$(I):ABT$=ABT$(I):TT=T(I):ZI
$=ZI$(I)
1210 A$(I)=A$(T):N$(I)=N$(T):ABT$(I)=ABT$(T)T
(I)=T(T):ZI$(I)=ZI$(T)
1220 A$(T)=A$:N$(T)=N$:ABT$(T)=ABT$:T(T)=TT:ZI
$(T)=ZI$:RETURN
```

Diskussion:

Dieses Programm unterscheidet sich von den Vorläuferlösungen durch eine sparsame Verwendung von Variablennamen. Die Minimierung der Variablennamen erleichtert dem Computer das Durchsuchen seines Variablenpools und es wirkte sich als Reduzierung der Laufzeit um volle 5 s je Sort aus.

Zeile 110: Zum Durchsuchen auf kleinsten Sortierbegriff muß das Programm sich einen größten Sortierbegriff "merken" gegen den es vergleichen kann. Alternativ zu dem Verfahren einer Zuweisung des größtmöglichen Sortierbegriffes ist es möglich, den ersten gelesenen Sortierbegriffes des gelesenen Quellenfeldes zum Vergleich heranzuziehen.

Zeile 1200 ff: In diesem Unterprogramm findet der Austausch statt. Das Tauschverfahren erinnert an den Bubblesort, jedoch ist hier die Entfernung der beiden auszutauschenden Positionen größer.

Zeilen 1000 ff: Auch das Druckunterprogramm ist verändert worden. Die Abfrage auf das Datenende fehlt und der Ausstieg erfolgt auf jeden Fall über das Ende der Schleife in Zeile 1040. Die beiden Leerzeilen werden durch einen einzigen PRINT-Befehl erzeugt.

Übung:
Stellen Sie bitte die Laufzeit der Abschnitte auch dieses Sortier-Programmes fest !

Zusammenfassung der Ergebnisse:

Programmabschnitt	Bubblesort	Einfügen	Auswahl	Auswahl red. Variable
Plausibilierung	8 s	52 s	8 s	7 s
Sort nach Namen	157 s		34 s	29 s
Sort nach Abteilung	156 s	43 s	34 s	29 s
Dauer (ohne Druck)	321 s	95 s	76 s	65 s
	495 %	146 %	117 %	100 %

Das Ergebnis läßt sich so zusammenfassen:
Das Bubblesortverfahren ist sehr langsam und sollte deshalb vermieden werden.
Das Sortieren durch Auswählen ist dem Sortieren durch Einfügen vorzuziehen. Vermutlich sind Kartenspieler beim Preisskat der gleichen Ansicht, wenn auch aus anderen Gründen ???

Wie geht es weiter

Wir beenden unsere 2. Fallstudie durch Ergänzen der beiden noch fehlenden Sortierungen nach Telefonnummer und Zimmernummer. Dabei werden wir den ASCII-Code kennenlernen.

Die dritte Fallstudie wird uns zum Thema der Simulation eines technischen Vorganges führen. Wir werden versuchen, daß Programm zur dritten Fallstudie strukturiert zu entwickeln.

Der Textteil der nächsten Lektion ist sehr umfangreich. Jedoch lassen sich weite Passagen des Textes ziemlich rasch und oberflächlich lesen.

NOTIZEN

9. Lektion

9. LEKTION

In unserer 2. Fallstudie haben wir unter den "einfachen" Sortierverfahren das Sortieren durch Auswählen als für unsere Anwendung am besten geeignet ermittelt.

Unsere Lösung, die alle vier in der Aufgabenstellung geforderten Sortierfolgen beinhaltet, ist nachfolgend abgedruckt:

```
1 CLEAR 5000
10 CLS:PRINT"TELEFONLISTE, SORTIERT NACH VERSC
HIEDENEN BEGRIFFEN
15 DEFINT A-Z
20 T=140:DIM A$(T),N$(T),ABT$(T),T(T),ZI$(T):R
EM T=ANZAHL MITARBEITER
30 REM A$(T)=ANREDE, N$(T)=NAME, ABT$(T)=ABTEI
LUNG, T(T)=TELEFONNR.,ZI$(T)=ZIMMERNR.

35 REM ERSTES EINLESEN DABEI PLAUSI DER DATEN.
40 PRINT"EINLESEN UND PLAUSIBILIEREN":J=T:FOR
I=1 TO J:PRINT@960,I;:
READ A$(I):IF A$(I)="9999"THEN Z=I-1:I=J+1:GO
TO 80:REM DATENENDE
50 IF A$(I)<>"" AND A$(I)<>"F"AND A$(I)<>"H" A
ND A$(I)<>"FRL"
THEN PRINTA$;"ANREDE ("2000+10*I;") UNZULAESS
IG":INPUT A$
55 READ N$(I),ABT$(I),T(I),ZI$(I)
60 T=T(I):IF T<200 OR T>299 THEN IF NOT T=7 AN
D NOT T=9 AND NOT T=111
AND NOT T=112 PRINTT(I);" TELEFONNR. ("2000+10*
I;") Unzulaessig.":INPUT A$
```

```
70 ZI$=ZI$(I):IF ZI$="EMPFANG" OR ZI$="EXTERN"
THEN 80
72 T=VAL(ZI$):IF T>=10 AND T<=20
OR T>=100 AND T<=120
OR T>=200 AND T<=220
OR T>=300 AND T<=320 THEN 80
75 PRINTZI$;" ZIMMERNR. ("2000+I*10;") UNZULAE
SSIG":INPUT A$
80 NEXT

100 REM 1.SORT: NAME, ABTEILUNG. N$ U. ABT$= Z
WISCHENSPEICHER
KLEINSTE ELEMENTE
110 CLS:PRINT"SORT NACH NAMEN":FOR I=1 TO Z-1:
PRINT@960,I;:
N$="ZZZZZ":ABT$=N$
120 FOR J=I TO Z:IF N$(J)<N$ THEN N$=N$(J):ABT
$=ABT$(J):T=J:GOTO 130
125 IF N$(J)=N$ THEN IF ABT$(J)<ABT$ ABT$=ABT$
(J):T=J
130 NEXT J:GOSUB 1200:NEXT I
170 GOSUB 1000:REM AUSDRUCK

200 REM 2.SORT: ABTEILUNG, NAME. ABT$ U. N$= Z
WISCHENSPEICHER
KLEINSTE ELEMENTE
210 CLS:PRINT"SORT NACH ABT.":FOR I=1 TO Z-1:P
RINT@960,I;:N$="ZZZZZ":
ABT$=N$
220 FOR J=I TO Z:IF ABT$(J)<ABT$ THEN N$=N$(J
):ABT$=ABT$(J):T=J:GOTO230
225 IF ABT$(J)=ABT$ THEN IF ABT$(J)<ABT$ N$=N$
(J):T=J
230 NEXT J:GOSUB 1200:NEXT I
270 GOSUB 1000:REM AUSDRUCK
300 REM 3.SORT: ZIMMER, NAME, ABT. ZI$, N$ U.
ABT$=ZWISCHENSPEICHER
KLEINSTE ELEMENTE
310 CLS:PRINT"SORT NACH ZIMMERNR.":FOR I=1 TO
Z-1:PRINT@960,I;:ZI$="ZZZZZ":
N$=ZI$:ABT$=ZI$
320 FOR J=I TO Z:IF ZI$(J)<ZI$ THEN ZI$=ZI$(J)
:N$=N$(J):ABT$=ABT$(J):T=J
```

```
330 IF ZI$(J)=ZI$ THEN IF N$(J)<N$ N$=N$(J):AB
T$=ABT$(J):T=J
ELSE IF N$(J)=N$ THEN IF ABT$(J)<ABT$ ZI$=ZI$(
J):ABT$=ABT$(J):T=J
340 NEXT J:GOSUB 1200:NEXT I
370 GOSUB 1000:REM AUSDRUCK
400 REM 4.SORT: TELEFONNR., NAME, ABT. U, N$
U. ABT$=ZWISCHENSPEICHER
KLEINSTE ELEMENTE
410 CLS:PRINT"SORT NACH TELEFONNR.":FOR I=1 TO
 Z-1:PRINT@960,I;:U=9999:
N$="ZZZZZ":ABT$=N$
420 FOR J=I TO Z:IF T(J)<U THEN U=T(J):N$=N$(J
):ABT$=ABT$(J):T=J
430 IF T(J)=U THEN IF N$(J)<N$ N$=N$(J):ABT$=A
BT$(J):T=J
ELSE IF N$(J)=N$ THEN IF ABT$(J)<ABT$ U=T(J):A
BT$=ABT$(J):T=J
440 NEXT J:GOSUB 1200:NEXT I
470 GOSUB 1000:REM AUSDRUCK
480 END

999 REM AUSGABE DER LISTE
1000 PRINT"TELEFONVERZEICHNIS DER KLINGELDREHE
R A.G., STAND: ";D$
1010 PRINT:PRINT"NAME","ABTEILUNG","TELEFON","
ZIMMER"
1020 FOR I=1 TO Z
1025 IF A$(I)="" THEN A$="     " ELSE A$=LEFT$(
A$(I)+"."+"    ",4)
1030 N$=N$(I):ABT$=ABT$(I):T=T(I):ZI$=ZI$(I)
1032 IF LEN(N$)>15 THEN N$=LEFT$(N$,15))
1034 IF LEN(ABT$)>15 THEN ABT$=LEFT$(ABT$,15)
1040 PRINTA$;N$,ABT$,T,ZI$:NEXT:PRINT"

":RETURN

1199 REM  VERTAUSCHEN MIT ZWISCHENBEREICH
1200 A$=A$(I):N$=N$(I):ABT$=ABT$(I):TT=T(I):ZI
$=ZI$(I)
1210 A$(I)=A$(T):N$(I)=N$(T):ABT$(I)=ABT$(T):T
(I)=T(T):ZI$(I)=ZI$(T)
```

```
1220 A$(T)=A$:N$(T)=N$:ABT$(T)=ABT$:T(T)=TT:ZI
$(T)=ZI$:RETURN
```

Diskussion:

Das Programmlisting ist bei den ersten beiden Sorts (nach Namen und nach Abteilung) etwas verändert worden: es fehlen zwei unnötige Zuweisungsbefehle in den Zeilen 125 und 225.

Beim Testlauf des Programmes fällt eine eigenartige Sortierfolge beim Sort nach Raumnummern auf. So interpretiert das Programm: Zimmernummer 120 ist kleiner als 15.

Da die Zimmernummern als Strings verarbeitet werden, sieht das Programm 120 und 15 nicht als Zahl an, sondern als Bytes.

Um eine richtige Sortierfolge erstellen zu können, müssen wir die Angelegenheit aus dem Blickwinkel des Programmes sehen und uns Gedanken über den Inhalt der betreffenden Strings machen.

Die Strings sind über die Tastatur bei den DATA-Zeilen ins Programm gelangt.
Es gibt in unserem BASIC zwei Befehle, die uns weiterhelfen können, herauszufinden, wie das über die Tastatur eingegebene Byte aussieht:

INKEY$

INKEY$ ermöglicht die Eingabe eines Zeichens, ohne daß dabei die NEW-LINE-Taste gedrückt werden muß. Beispiel dazu: siehe nächste Übung. Das eingelesene Zeichen ist ein 1 Byte langer String.

ASC(string)

Diese Funktion ermittelt das Bitmuster des ersten Zeichens der angegebenen Zeichenkette.
ASC erinnert an das Wort "ASCII", auf das unmittelbar hinter der nächsten Übung genauer eingegangen wird.

Da der Funktionsname ASC an ASCII erinnert, wird die Funktion ASC mit Vorliebe in BASIC-Handbüchern falsch beschrieben: Genau wie auch Strings nicht auf Zeichen des ASCII-Codes beschränkt sind, ist auch die Funktion ASC nicht auf ASCII-Zeichen beschränkt.

Übung:
Bitte, lassen Sie das folgende kleine Programm laufen und betätigen Sie die Tasten der Tastatur. Hinweis: RUN 3000

```
3000 A$=INKEY$: IF A$="" THEN 3000 ELSE ?A$;"=
"ASC(A$):A$="":GOTO 3000
```

Das Programm zeigt bei jeder Taste, die Sie drücken, eine Art Übersetzung. Der Übersetzungscode ist definiert als

ASCII-Code

CHR$(num)

Der Befehl CHR$ erzeugt aus einem Bitmuster im Bereich 0 bis 255 ein Zeichen. CHR$(num) ist somit die Umkehrung von ASC(string).

Wir können diesen Befehl dazu nutzen, uns eine Codetabelle des ASCII-Codes anzufertigen:

Übung:

```
3000 CLS:FOR I=0 TO 255:?I;"="CHR$(I),:NEXT
```

Ergebnis:
Läßt man dieses Programm laufen, so erhält man ein lebhaftes Bild, denn die Steuerzeichen des Codes (0 bis 31) bringen die Anzeige durcheinander.
Wir wollen dieses Programm deshalb auf die anzeigbaren Zeichen abändern, indem wir die Schleife auf den Bereich 32 bis 191 beschränken.

Mit der nachfolgend abgedruckten Version des Programmes können Sie sich eine Codetabelle auf einem Drucker anfertigen:

```
3000 FOR I=32 TO 191 STEP 9
3010 FOR J=I TO I+8:LPRINT "     "CHR$(J);"="US
ING"###";J;:NEXT J
3020 LPRINT:NEXT I
```

Durch LPRINT wird die Ausgabe auf den Drucker gegeben. Davon abgesehen arbeitet LPRINT ähnlich wie PRINT.

Die offizielle Codetabelle für den Video-Genie steht im BASIC-Manual APPENDIX C.
Die offizielle Codetabelle für den TRS-80 steht im Handbuch für BASIC Level II, ebenfalls im Anhang C.

Leider fehlt im BASIC Manual des Video Genie ein Hinweis auf die Verwendung der Codes 192 bis 255, die bei unserer Übung den Aufbau des Schirmes so durcheinandergebracht haben:

Die Codes 192 bis 255 sind als Platzsparcode für Tabulatormarken für 0 bis 63 Leerstellen vorgesehen – daher auch die Änderung des Schirmbildes !

Die Tabulation über Platzsparcodes

Wenn ein String mit PRINT angezeigt wird und er Platzsparcodes für die Tabulation enthält (192 bis 255), so wird die Tabulation auf die Stelle

Platzsparcode -- 192

ausgeführt.

Übung (im Direktmodus):

```
CLS:FOR T=192 TO 207:?CHR$(T);"TAB AUF STELLE"
T-192:NEXT
```

Diese Art der Tabulation ist eine Besonderheit des im Video Genie und TRS-80 installierten BASIC-Dialektes. Sie ist eine solche Besonderheit, daß man sie im BASIC-Manual des Video Genie glatt verschwiegen hat...

Abschluß der 2. Fallstudie:

Beim Sortieren nach Zimmernummern bewertet das Programm die Strings ZI$ unmittelbar. Somit ist z. B. "15" größer als "120", denn ein Stringvergleich geht immer von links nach rechts, und die "5" ist eindeutig größer als die "2" an der entsprechenden Stelle des Strings "120".

Merke

Sortieren von Strings mit numerischen Inhalt:
Die Sortierfolge ist nur numerisch, wenn die Strings mit numerischem Inhalt gleich lang sind, d. h., die gleiche Stellenanzahl haben und ausschliesslich Ziffern enthalten.

In unserem Fall ist die Änderung relativ geringfügig. Das Programm ist nachfolgend abgedruckt. Es zeigt beim Datum allerdings nichts an (Druckunterprogramm, String D$). Es wäre sehr gut, wenn Sie sich zur nächsten Lektion eine Erweiterung des Programmes ausdenken würden, mit der man das Datum ergänzen kann.

```
1 CLEAR 5000
10 CLS:PRINT"TELEFONLISTE, SORTIERT NACH VERSC
HIEDENEN BEGRIFFEN
15 DEFINT A-Z
20 T=140:DIM A$(T),N$(T),ABT$(T),T(T),ZI$(T):R
EM T=ANZAHL MITARBEITER
30 REM A$(T)=ANREDE, N$(T)=NAME, ABT$(T)=ABTEI
LUNG, T(T)=TELEFONNR.,
ZI$(T)=ZIMMERNR.
35 REM ERSTES EINLESEN DABEI PLAUSI DER DATEN.
40 PRINT"EINLESEN UND PLAUSIBILIEREN":J=T:FOR
I=1 TO J:PRINT@960,I;:
READ A$(I):IF A$(I)="9999"THEN Z=I-1:I=J+1:GO
TO 80:REM DATENENDE
```

```
50 IF A$(I)<>"" AND A$(I)<>"F"AND A$(I)<>"H" A
ND A$(I)<>"FRL"
THEN PRINTA$;"ANREDE ("2000+10*I;") UNZULAESSI
G":INPUT A$
55 READ N$(I),ABT$(I),T(I),ZI$(I)
60 T=T(I):IF T<200 OR T>299 THEN IF NOT T=7 A
ND NOT T=9 AND NOT T=111
AND NOT T=112 PRINTT(I);" TELEFONNR. ("2000+10*
I;") Unzulaessig.":INPUT A$
70 ZI$=ZI$(I):IF ZI$="EMPFANG" OR ZI$="EXTERN"
THEN 80
72 T=VAL(ZI$):IF T>=10 AND T<=20
OR T>=100 AND T<=120
OR T>=200 AND T<=220
OR T>=300 AND T<=320 THEN 80
75 PRINTZI$;" ZIMMERNR. ("2000+I*10;") UNZULAE
SSIG":INPUT A$
80 NEXT

100 REM 1.SORT: NAME, ABTEILUNG. N$ U. ABT$= Z
WISCHENSPEICHER
KLEINSTE ELEMENTE
110 CLS:PRINT"SORT NACH NAMEN":FOR I=1 TO Z-1:
PRINT@960,I;:
N$="ZZZZZ":ABT$=N$
120 FOR J=I TO Z:IF N$(J)<N$ THEN N$=N$(J):ABT
$=ABT$(J):T=J:GOTO 130
125 IF N$(J)=N$ THEN IF ABT$(J)<ABT$ ABT$=ABT$
(J):T=J
130 NEXT J:GOSUB 1200:NEXT I
170 GOSUB 1000:REM AUSDRUCK

200 REM 2.SORT: ABTEILUNG, NAME. ABT$ U. N$= Z
WISCHENSPEICHER
KLEINSTE ELEMENTE
210 CLS:PRINT"SORT NACH ABT.":FOR I=1 TO Z-1:P
RINT@960,I;:N$="ZZZZZ":
ABT$=N$
220 FOR J=I TO Z:IF ABT$(J)<ABT$ THEN N$=N$(J)
:ABT$=ABT$(J):T=J:GOTO230
225 IF ABT$(J)=ABT$ THEN IF ABT$(J)<ABT$ N$=N$
(J):T=J
```

```
230 NEXT J:GOSUB 1200:NEXT I
270 GOSUB 1000:REM AUSDRUCK
300 REM 3.SORT: ZIMMER, NAME, ABT. ZI$, N$ U.
ABT$=ZWISCHENSPEICHER
KLEINSTE ELEMENTE
310 CLS:PRINT"SORT NACH ZIMMERNR.":FOR I=1 TO Z
-1:PRINT@960,I;:ZI$="ZZZZZ":
N$=ZI$:ABT$=ZI$
320 FOR J=I TO Z:ZJ$=RIGHT$("000"+ZI$(J),3):IF
ZJ$<ZI$ THEN
ZI$=ZJ$:N$=N$(J):ABT$=ABT$(J):T=J
330 IF ZJ$=ZI$ THEN IF N$(J)<N$ N$=N$(J):ABT$=
ABT$(J):T=J
ELSE IF N$(J)=N$ THEN IF ABT$(J)<ABT$ ZI$=ZJ$:
ABT$=ABT$(J):T=J
340 NEXT J:GOSUB 1200:NEXT I
370 GOSUB 1000:REM AUSDRUCK
400 REM 4.SORT: TELEFONNR., NAME, ABT. U, N$
U. ABT$=ZWISCHENSPEICHER
KLEINSTE ELEMENTE
410 CLS:PRINT"SORT NACH TELEFONNR.":FOR I=1 TO
Z-1:PRINT@960,I;:U=9999:
N$="ZZZZZ":ABT$=N$
420 FOR J=I TO Z:IF T(J)<U THEN U=T(J):N$=N$(J
):ABT$=ABT$(J):T=J
430 IF T(J)=U THEN IF N$(J)<N$ N$=N$(J):ABT$=A
BT$(J):T=J
ELSE IF N$(J)=N$ THEN IF ABT$(J)<ABT$ U=T(J):A
BT$=ABT$(J):T=J
440 NEXT J:GOSUB 1200:NEXT I
470 GOSUB 1000:REM AUSDRUCK
480 END

999 REM AUSGABE DER LISTE
1000 PRINT"TELEFONVERZEICHNIS DER KLINGELDREHE
R A.G., STAND: ";D$
1010 PRINT:PRINT"NAME","ABTEILUNG","TELEFON","
ZIMMER"
1020 FOR I=1 TO Z
1025 IF A$(I)="" THEN A$="     " ELSE A$=LEFT$(
A$(I)+"."+"    ",4)
1030 N$=N$(I):ABT$=ABT$(I):T=T(I):ZI$=ZI$(I)
1032 IF LEN(N$)>15 THEN N$=LEFT$(N$,15))
```

```
1034 IF LEN(ABT$)>15 THEN ABT$=LEFT$(ABT$,15)
1040 PRINTA$;N$,ABT$,T,ZI$:NEXT:PRINT"

":RETURN

1199 REM  VERTAUSCHEN MIT ZWISCHENBEREICH
1200 A$=A$(I):N$=N$(I):ABT$=ABT$(I):TT=T(I):ZI
$=ZI$(I)
1210 A$(I)=A$(T):N$(I)=N$(T):ABT$(I)=ABT$(T):T
(I)=T(T):ZI$(I)=ZI$(T)
1220 A$(T)=A$:N$(T)=N$:ABT$(T)=ABT$:T(T)=TT:ZI
$(T)=ZI$:RETURN
```

Die 2. Fallstudie ist hiermit abgeschlossen.

Bevor wir die 3. Fallstudie beginnen, wollen wir uns vornehmen, bei der 3. Fallstudie anders vorzugehen:

nämlich: strukturiert

Doch zunächst einige Überlegungen zum Thema "Struktur".

Was ist eigentlich eine strukturierte Vorgehensweise bei der Erstellung von Software ?

Einige große Worte vorweg:

Strukturierung

An den Stil eines Programmes werden Anforderungen gestellt:

1. Ein Programm muß für einen sachkundigen Dritten interpretierbar sein.
2. Ein Programm muß für einen sachkundigen Dritten selbstdokumentierend sein. Der Programmablauf und das verwendete Programmierverfahren muß sichtbar und verständlich sein.
3. Die Programmodule müssen unabhängig voneinander testbar sein.

Als "Musterknabe" der modernen Programmiersprachen gilt die Programmiersprache "PASCAL". Der Programmiersprache "BASIC" wird

häufig der Vorwurf gemacht, daß sich bei größeren Programmen die oben genannten drei Anforderungen nicht realisieren lassen.

Es gibt jedoch unterschiedliche BASIC-Dialekte, und wir wollen einmal die inneren Strukturierungsmöglichkeiten von PASCAL den inneren Strukutierungsmöglichkeiten des im Video Genie bzw. TRS-80 realisierten BASIC-Dialektes gegenüberstellen.

A. Die FOR-Schleife

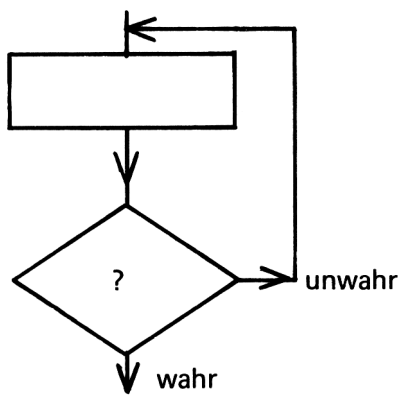
Die Schleife wird angewendet, wenn die Zahl der Durchläufe bereits vor Einstieg in die Schleife bekannt ist.

```
PASCAL: FOR /variable/ := /wert/ to /endwert/
DO /anweisung/ ;

BASIC : FOR  /variable/ =  /wert/ TO /endwert/
: /anweisung/:NEXT /wert/
```

B. REPEAT ... UNTIL

Diese Anweisungsfolge wird verwendet, wenn beim Einstieg in die Schleife die Zahl der Durchgänge noch nicht festliegt. Dies ist bei interativen Verfahren der Fall, deshalb wird die Anweisungsfolge auch gelegentlich "Iteration" genannt.



```
PASCAL: REPEAT /anweisung/ UNTIL /bedingung/ ;

BASIC: /zeilennr.1  anweisung/ :IF NOT
/bedingung/ THEN /zeilennr.1/
```

Auch hier ist der Unterschied zwischen PASCAL und BASIC nicht wesentlich. Sofern man die Verwendung des THEN-Befehles unbedingt vermeiden möchte, gibt es ein zweites Verfahren mit FOR-TO und Schrittweite Null. Dieses Verfahren sei an einem Beispiel gezeigt:

```
100 FOR I=0 TO 1 STEP 0:/anweisung/:IF
/bedingung/ THEN I=I+1
110 NEXT I
```

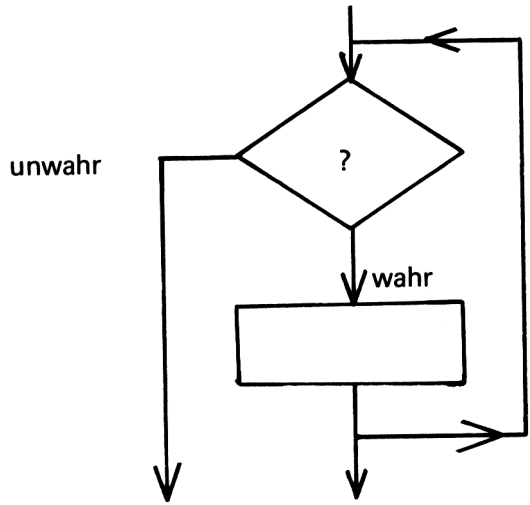
Eine noch raffiniertere Version dieses Verfahrens arbeitet mit der internen Darstellung von "wahr" und "unwahr" zusammen: Bei ihr ist jedoch keine Kompabilität zu anderen BASIC-Dialekten gewährleistet:

```
100 FOR I=0 TO -1 STEP -1: /anweisung/ :I=
/bedingung/:NEXT I
```

Nun ja, wenn Ihnen so etwas einmal in einem Programm auffällt, wissen Sie ab heute, was es bewirken soll(te ?).

C. WHILE . . . DO

Diese Anweisung stellt eine Ablaufänderung gegenüber der REPEAT . . . UNTIL – Anweisung dar. Bei REPEAT . . . UNTIL wird die Anweisung mindestens einmal ausgeführt, bei WHILE . . . DO wird sie unter Umständen keinmal ausgeführt:



Dieser kleine Unterschied wird eigentlich bei Betrachtung der BASIC-Codierung klarer als bei Betrachtung der PASCAL-Codierung:

```
PASCAL: WHILE /bedingung/ DO /anweisung/ ;

BASIC: /zeilennr.1  IF /bedingung/ THEN
/anweisung/:GOTO /zeilennr.1/
```

D. IF ... THEN ... ELSE

```
PASCAL: IF /bedingung/ THEN /anweisung 1/
ELSE /anweisung 2/ ;

BASIC:  IF /bedingung/ THEN /anweisung 1/
ELSE /anweisung 2/
```

E. CASE

Diese Anweisung führt eine Multiweg-Entscheidung durch:

PASCAL:

```
CASE  /variable/ OF
        /wert/ : /anweisung/ ;
        /wert/ : /anweisung/ ;
        /wert/ : /anweisung/
END;
```

BASIC:

```
IF /bedingung/ THEN /anweisung/
IF /bedingung/ THEN /anweisung/
IF /bedingung/ THEN /anweisung/
```

Auch hier ist die Ähnlichkeit zwischen PASCAL und BASIC frappierend.

> Der obige Abschnitt reicht zu einem wertenden Vergleich beider Programmiersprachen nicht aus, denn PASCAL bietet einige Dinge die der im Video Genie und TRS-80 realisierte BASIC-Dialekt nicht bietet und die in diesem Kursus auch nicht besprochen werden. Eine Nebenabsicht dieses Abschnittes ist es jedoch, Sie als Kursusteilnehmer gegenüber in Fachartikeln angestellten Vergleichen "PASCAL contra BASIC" kritischer zu machen: "Das BASIC" gibt es gar nicht, der Autor eines solchen Fachartikels meint fast immer einen bestimmten BASIC-Dialekt. Der Leser eines solchen Fachartikels muß deshalb kritisch prüfen welche dem BASIC nachgesagten Eigenschaften für "seinen" BASIC-Dialekt zutreffen.

Wesentlicher als die inneren Programmstrukturen ist die Vorgehensweise bei der Erstellung von Programmen. Eine iterative Programmierung, wie wir sie bei der 1. und 2. Fallstudie durchgeführt haben, ist bei einer Realisierung mit PASCAL undenkbar:

Bei PASCAL muß das gesamte Programm auf Papier fertiggeschrieben und strukturiert sein, bevor das Eintippen des Programmes am Computer erfolgen kann. PASCAL zwingt den Programmierer dazu, in einem TOP-DOWN-Design von der Gesamtlösung ausgehend Programm-

module zu definieren. Ist das geschehen, erfolgt die Programmierung als DOWN-TOP-Design: Es werden zuerst die Programmodule programmiert und erst am Ende das den Ablauf bestimmende Hauptmodul.

Diese Art der Realisierung hat bei größeren Programmen wesentliche Vorteile, stellt aber auch wesentlich größere Anforderungen an den Programmierer. Die iterative Programmierung ist – überspitzt gesagt – das Programmierverfahren des "Bastlers", die bei PASCAL erzwungene Vorgehensweise ist dagegen ingenieurgerecht.

Bevor wir die 3. Fallstudie beginnen, wollen wir alle die uns noch fehlenden Befehle besprechen und damit auch das Erlernen von Befehlen beenden.

Direktspeicherzugriffsbefehle PEEK und POKE

PEEK und POKE werden gerne verwendet, um fehlende BASIC-Funktionen zu ergänzen.

> Die Häufigkeit von PEEK- und POKE-Befehlen in einem BASIC-Programm ist ein Mass für die Güte des Befehlsvorrates des verwendeten BASIC-Dialektes. Je häufiger PEEK und POKE vorkommt, desto schlechter paßt der betreffende Befehlsumfang des BASIC-Dialektes zur Aufgabenstellung.
>
> – – – – – – – – – – –
>
> In Fachzeitschriften abgedruckten Programme mit PEEK und POKE soll man meiden wie die Pest, denn sie sind fast nie auf andere Systeme übertragbar. Sofern in den Titeln solcher Artikel nicht das System mit angegeben ist, sollte man an die Redaktionen empörte Leserbriefe schreiben.

Die BASIC-Dialekte des Video Genies und des TRS-80 sind hervorragend angepaßt, so daß man PEEK und POKE so gut wie nicht benötigt. Eine kleine Ausnahme ist die Druckerunterstützung, bei der es eine sinnvolle Anwendung für PEEK und POKE gibt.

PEEK und POKE	ermöglichen Direktspeicherzugriffe.
PEEK (adresse)	liest vom Speicher nach BASIC.
POKE adresse,wert	schreibt von BASIC aus in den Speicher.

Die Adresse kann im BASIC ohne Diskette nur dezimal angegeben werden. Unter Disketten-BASIC ist auch eine hexadezimale Angabe möglich.

Es gilt:
Adresse 8000 Hexa bis FFFF Hexa = –32768 bis –1
Adresse 0000 Hexa bis 7FFF Hexa = 0 bis 32767

Der Übergang ist also:

7FFF Hexa	= 32767
8000 Hexa	= –32768

Der Video-RAM und PEEK

Die nachfolgende Routine nutzt den PEEK-Befehl, um den angezeigten Bildschirminhalt auszudrucken:
Die Adresse des Bildschirmwiederholungsspeichers beginnt mit 3C00 Hexa. Jede Zeile ist 64 Zeichen breit.

```
50000 FOR I=0 TO 15:FOR J=0 TO 63
50001 LPRINT CHR$(PEEK(15360+I*64+J));:NEXT J:
LPRINT:NEXT I
```

Der einzeln stehende LPRINT-Befehl in Zeile 50001 erzeugt einen Zeilenvorschub am Zeilenende.

Drucker-DCB

Beim Video Genie und TRS-80 werden folgende Peripheriegeräte über Gerätesteuerblöcke (DCB) bedient:

Tastatur
Bildschirm
Drucker

Die Aufbauten sind:

	Tastatur	Bildschirm	Drucker
Startadresse	16405	16413	16421
Startadresse +			
0	0=DCB-Typ	0=DCB-Typ	0=DCB-Typ
1	LSB Treiberadr.	LSB Treiberadr.	LSB Treiberadr.
2	MSB Treiberadr.	MSB Treiberadr.	MSB Treiberadr.
3	0	LSB Cursorpos.	Zeilen/Seite
4	0	MSB Cursorpos.	Zeilenzähler
5	0	Cursorzeichen	0
6	ASCII K	ASCII D	ASCII P
7	ASCII I	ASCII O	ASCII R

Mit den DCB's kann man eine ganze Menge anfangen, insbesondere der Peripherie andere Eigenschaften verleihen.

Die Vielfalt der Möglichkeiten sei am Beispiel von Listings für dieses Buch beschrieben:
Der Drucker, mit dem teilweise die Listings gedruckt wurden, stammt von dem Hersteller HEATHKIT und ist über eine serielle Schnittstelle RS-232-C Schnittstelle angeschlossen: Änderung des Drucker-DCB's. Es wird Kleinschrift verwendet: Änderung des Tastatur-DCB's und Änderung des Bildschirm-DCB's sowie eine Hardware-Änderung des Bildschirmwiederholungsspeichers.

Den Drucker-DCB wollen wir genauer betrachten, denn aus ihm lassen sich Anforderungen an den Drucker ableiten, die man beim Kauf eines Druckers für den TRS-80 / Video Genie berücksichtigen sollte:

1. Der DCB gibt das Steuerzeichen
   LF Zeilenvorschub Code: 10
   nicht aus, sondern er wandelt es um in
   CR Wagenrücklauf Code: 13

Deshalb muß der Drucker ohne LF auskommen.

Der Druck-DCB entscheidet selbst, wann ein Seitenvorschub erforderlich ist. Ein Seitenvorschub kann erzwungen werden durch den Befehl

LPRINT CHR$(12)

Adresse 16424 enthält die Anzahl Zeilen einer vollen Seite.
Adresse 16425 die Anzahl bereits gedruckter Zeilen der laufenden Seite.

Übung:
Ändern Sie die Anzahl Zeilen pro Seite

Lösung:

```
?PEEK (16424)          67
POKE 16424,60
?PEEK (16424)          60
```

Übung:
Stellen Sie fest, wieviel Zeilen auf der gerade gedruckten Seite noch frei sind:

```
?PEEK(16424)–PEEK(16425)   60
```

ON . . . GOTO und ON . . . GOSUB

Diese beiden Befehle führen eine Multiweg-Verzweigung oder einen Multiweg-Unterprogrammaufruf durch.

Der Verzweigungsmechanismus beinhaltet eine Plausibilierung des Wertes: ist der Wert unzulässig, so erfolgt keine Verzweigung. Der Wert wird als ganzzahlig betrachtet, d. h., etwaigige Nachkommastellen werden abgeschnitten.

Eine typische Anwendung ist gegeben bei der Menue-Anzeige:
Das Programm stellt einige Alternativen zur Auswahl und verzweigt zur gewählten Alternative:

```
10 REM DIES PROGRAMM DEMONSTRIERT DIE ON...GOT
O FUNKTION
20 REM
30 CLS
40 PRINT"BITTE EINGEBEN:
1 FUER QUADRATWURZEL
2 FUER LOG
3 FUER SIN
4 FUER COS
```

```
50 A$=INKEY$:IF A$="" THEN 50

60 ON VAL(A$) GOTO 100, 200, 300, 400
70 PRINT"SIE SOLLTEN NICHT ´´";A$;"´´, SONDERN
EINE ZIFFER VON 1...4 EINGEBEN !
VERSUCHEN SIE ES NOCHMAL":GOTO 50

100 INPUT"AUS WELCHEM WERT SOLL DIE WURZEL GE
ZOGEN WERDEN
";A
110 PRINT"IHR ERGEBNIS="SQR(A)
120 GOTO900

200 INPUT"DEN LOG ZU WELCHEM WERT
";A
210 PRINT"LN("A;")=";LOG(A);"
LOG("A;")="LOG(A)*.43329
220 GOTO900

300 INPUT"DEN SIN ZU WELCHEM WINKEL (RAD)
";A
310 PRINT"SIN("A;")="SIN(A)
320 GOTO900

400 INPUT"DEN COS ZU WELCHEM WINKEL (RAD)
";A
410 PRINT"SIN("A;"="SIN(A)

900 INPUT"
=====> NEW LINE";A$
910 RUN
920 END
```

Wenn Sie bei dem eben abgedruckten Programm aus dem ON . . . GOTO in Zeile 60 ein ON . . . GOSUB machen, stellen Sie fest, daß die Rückehr aus dem Unterprogramm in die ON . . . GOSUB-Zeile hinein unmittelbar nach Ausführung des Unterprogrammes erfolgt. Zeilen 120, 220, 320 und die neue Zeile 420 muß dann den Inhalt RETURN haben.

Input

Dieser Befehl ist für uns ein "Oldtimer", denn wir wenden ihn schon

seit geraumer Zeit an.
Wir nehmen ihn uns dennoch vor, um auf etwas hinzuweisen:

Übung:

```
10 INPUT"WERT";A
20 PRINT A
30 INPUT"STRING";A$
40 ?A$;"LAENGE=";LEN (A$)
50 GOTO 10
```

Bitte lassen Sie dieses Programm laufen und geben Sie nacheinander ein:

Eingabe:	Ergebnis:
WERT: 10	10
STRING: AAA	AAA LAENGE 3
WERT: NEW LINE	10
STRING: NEW LINE	AAA LAENGE 3

Sie sehen, die alleinige Eingabe von NEW LINE läßt den alten Inhalt unverändert.
Dieses Verhalten ist bei anderen BASIC-Dialekten u. U. anders und sollte – da es leider nicht immer in den Systemhandbüchern beschreiben wird – ausprobiert werden. Es gibt auch BASIC-Interpreter, die das Programm abbrechen, wenn in eine numerische Variable ein Buchstabe eingegeben wird – beim Video Genie und TRS-80 ist das nicht der Fall, der BASIC-Interpreter verhält sich hier absolut gutmütig.

Rechen(un)genauigkeiten

Alle mathematischen Funktionen werden nur in einfacher Genauigkeit durchgeführt – hier ist der BASIC-Dialekt des Video-Genie bzw. TRS-80 ausgesprochen schlecht.

Besonders peinlich ist z. B. folgende Ungenauigkeit:

Im Befehlsmodus:
?100.1–100 Ergebnis: .09999985

PRINT USING

Als Ausgleich für die Rechenungenauigkeiten gibt es die PRINT-USING-Funktion. Mit ihr ist es möglich, vorauszubestimmen, wieviele Stellen das Ergebnisses interessieren.
Dabei wird kaufmännisch gerundet.

Vereinfacht dargestellt wird eine Maske angegeben die aus #-Zeichen besteht:

```
?USING "###.##";100.1-100
```

Hier wird das Ergebnis mit 3 Stellen vor dem Dezimalpunkt und zwei Stellen hinter dem Dezimalpunkt ausgegeben. Führende Nullen werden hier unterdrückt !

Bei Scheckschreibung füllt man linksbündig gerne mit Sternchen auf, um das Verfälschen ausgedruckter Beträge zu erschweren.

Die Maske muß dann mit zwei ** anstelle der beiden ersten ##-Zeichen beginnen:

```
?USING"**##.##";1000.1-1000
```

> Der PRINT USING-Befehl stellt eine wesentliche Erleichterung bei der Ausgabe von Listen mit numerischen Werten dar. Sofern solche Applikationen geplant sind, ist die Unterstützung der PRINT USING-Funktion ein wesentliches Auswahlkriterium für einen Homecomputer.

Vorbereitung auf die 3. Fallstudie

Die dritte Fallstudie beschäftigt sich mit einer Simulation. Die Simulation in der dritten Fallstudie hat das Ziel, die Auswirkung der Reduzierung der Bremsverzögerung und der Anfahrbeschleunigung von Regalförderzeugen eines Hochregallagers auf die Umschlagleistung ermessen zu können.

Bei dieser Simulation benötigen wir eine Funktion zur Erzeugung von Zufallszahlen.

RANDOM

Dieser Befehl heißt in anderen BASIC-Dialekten häufig "RANDOMIZE". Wenn wir uns die Erzeugung der Zufallszahlen durch eine Lostrommel vorstellen, so mischt dieser Befehl die Lose gründlich durcheinander.

RND (wert)

Dieser Befehl liefert eine "unechte" Zufallszahl.
Ist Wert = 0, so liegt die Zufallszahl im Bereich von 0 und 1 und hat einfache Genauigkeit.
Ist Wert 〉 1, so werden ganzzahlige "unechte" Zufallszahlen zwischen 1 und dem Wert geliefert. RND(49) liefert Zahlen von 1 bis 49 und kann das Ausfüllen des Lottozettels erleichtern . . .

Andere BASIC-Dialekte arbeiten nur mit RND (0). Dann muß man die gefundene Zufallszahl multiplizieren, um sie in den gewünschten Bereich zu bringen.

Eine recht primitive Simulation ermittelt die Kreiszahl 3.14 . . .

```
10 CLS:PRINT"
ERMITTELN DER KREISZAHL AUS DEM VERGLEICH  DES
FLAECHENINHALTES EINES QUADRATES MIT DER
KANTENLAENGE 1 UND DES FLAECHENINHALTES
EINES VIERTELKREISSEGMENTES MIT DEM RADIUS 1


20 PRINT"VERFAHREN:
MIT EINEM ZUFALLSGENERATOR WERDEN FLAECHEN=
PUNKTE INNERHALB DES QUADRATES AUSGESUCHT.
 WENN SIE INNERHALB DES VIERTELKREISSEGMEN=
TES LIEGEN, GEHOEREN SIE ZUM VIERTELKREIS,
ANDERNFALLS LIEGEN SIE AUSSERHALB
30 INPUT"
======>NEW LINE";A$
```

```
40 CLS:DEFINT I-K:RANDOM
50 INPUT"WIE VIELE FLAECHENPUNKTE SOLLEN UNTER
SUCHT WERDEN
(100-30000)";I
60 IF I<100 OR I>30000 THEN PRINT"PFUI !": GO
TO 50

90 FOR J=1 TO I
100 X=RND(0):Y=RND(0):IF X*X+Y*Y> 1 THEN 110
ELSE K=K+1
110 PRINT@960,J,USING"##.#####";4*K/J;:NEXT
```

Aufgabenstellung zur 3. Fallstudie

Das zu untersuchende Hochregallager besteht aus 4 Gassen mit jeweils einem Regalförderzug.

Jede Gasse hat zwei Regalseiten, wobei ein Regalförderzeug beide Regalseiten bedient. Der Zugriff zu beiden Seiten ist gleich schnell.

Die Regalfächer jeder Seite sind nach folgenden Schema durchnummeriert:

```
00,12 ---------------------------------- 42,12
!                                          !
!                                          !
00,02                                   42,02
00,01                                   42,01
00,00 ---------------------------------- 42,00
```

Somit hat jede Gassenseite 43 horizontale Zeilen und 13 vertikale Etagen.

Folgende Fachadressen sind nicht benutzbar:

00,00 weil Übergabestation Einlagerbahn an Regalförderzeug
00,01 weil Übergabestation Regalfürderzeug an Auslagerbahn
00,02 weil reservierter Platz für die Feuerlöschpalette.

Die Paletten mit den einzelnen Produkten sind chaotisch über das gesamte Palettensilo verteilt.

Technische Daten:

Regalteilung	horizontal	1.35 m
	vertikal	2.10 m

Fahrleistung RFZ:

Fahrt	horizontal	2 m/s	vertikal	1 m/s
Beschleunigung		1 m/s*s		0.5 m/s*s
Bremsung		1 m/s*s		0.5 m/s*s

Gabelspieldauer: Palettenaufnahme: 20 s Palettenabsetzung: 24 s

Da zu häufig Kartons von den Paletten herabfallen, möchte man die Beschleunigungen beim Anfahren und Abbremsen auf

Beschleunigung	horizontal	0.8 m/s*s	vertikal	0.4 m/s*s
Bremsen		0.8 m/s*s		0.4 m/s*s

reduzieren.

Ihre Spezialaufgabe:

Stellen Sie fest, welche Zeit für 200 Auslagerungen benötigt werden. Stellen Sie fest, welche Zeit nach Reduzierung der Beschleunigung und der Abbremsung für 200 Auslagerungen benötigt werden.

Hinweis:
Der zuständige Lageringenieur hat es auf herkömmliche Art versucht, d. h., mit Formelbuch und Rechenschieber. Er sagt, daß es ihm nicht gelungen sei, einen vernünftigen Durchschnitt zu ermitteln und bittet Sie, mal eben ein Simulationsprogramm auf Ihrem Rechner einzugeben.
Außerdem wünscht er Ihnen dabei viel Erfolg !

10. Lektion

10. LEKTION

Das Programm, welches die Lösung unserer 3. Fallstudie bildet, hat einen wesentlich anderen Aufbau als alle anderen im Rahmen dieses Kursus erarbeiteten Programme.

Das Programm ist weiter hinten abgedruckt.

Man kann den Aufbau des Programmes wie folgt gliedern:

Zeilen 10 und 20: beschreiben, was das Programm macht.
Zeilen 30 bis 60: beschreiben den Aufbau des Programmes.
Zeilen 90 bis 150: bilden das Hauptprogramm. Diese Zeilen bestehen im wesentlichen aus GOSUB-Befehlen und sie bestimmen die Reihenfolge des Ablaufes der Module.

Abschließend folgen die Module: Sie alle sind als Unterprogramme codiert. Die Ansprungadressen haben durch 1000 ohne Rest teilbare Zeilennummern wie 3000, 4000, 5000, 6000 und 7000.

Der Aufbau eines jeden Moduls ist ähnlich gehalten:

Beispiel:
Modul Fachadresse auswählen, ab Zeile 1970:

Das Modul beginnt mit einer Überschrift. Darauf folgt eine Beschreibung der in diesem Modul verwendeten Variablen.
Das aufzurufende Unterprogramm (hier ab Zeile 2000) ist so geschrieben, daß es sich möglichst universell verwenden läßt. So hätte man bei der vorliegenden Aufgabenstellung die Zeilen 5020 und 5030

entfallen lassen und die Zeile 2000 einfacher schreiben können:

2000 H=RND(43)-1:V=RND(13)-1

Die im Programm konsequent praktizierte Trennung zwischen Daten und Befehlen hat jedoch Vorteile, denn sie erleichtern spätere Änderungen des Programmes: so kann man sicher sein, daß z. B. die Fachteilung nur in Zeile 5020 definiert ist – sie ist an keiner anderen Stelle des Programmes mehr definiert !

Nachteilig bei diesem Verfahren ist, daß der Programmentwurf komplizierter ist: die Gegenüberstellung der Zeile 2000 weiter oben zeigt diesen Effekt beispielhaft.

Ein weiterer Nachteil ist der große Speicherbedarf des Programmes der durch die vielen Kommentare hervorgerufen wird. Auch die Kommentare benötigen Laufzeit, so daß dieses Programm nicht sehr schnell ablaufen wird.

Ihre Aufgabe ist:

Entfernen Sie aus dem nachfolgenden Programmlisting möglichst viele Kommentare und geben Sie das Programm in den Rechner ein. Ihr so erstelltes "Miniprogramm" ist das auszuführende Programm, das abgedruckte Programm stellt Ihre getrennte Dokumentation dar.

Bitte achten Sie darauf, daß Sie die Zeilennummern beibehalten.

Machen Sie einen Probelauf und schreiben Sie das Laufergebnis auf und vergleichen Sie es mit den nachfolgenden Ergebnissen:

Beispiellösung:

Nr. des Laufes:	1. Durchlauf	2. Durchlauf
1	15455.2s	15595.5s
2	15199.5s	15619.9s
3	15636.0s	15486.3s
4	15515.8s	15710.3s

Zusammenfassung des Ergebnisses:
Das sanftere Anfahren und das sanftere Bremsen verändert die Fahrdauer so geringfügig, daß es im Betrieb nicht auffallen wird.

```
10 REM            R.F.M. 19.7.81
        SIMULATIONSPROGRAMM  KRANSPIEL
PGM UNTERSUCHT DIE AUSWIRKUNG VON AENDERUNGEN
DER FAHRWERTE (BESCHLEUNIGUNG, FAHRGESCHW.,
20 REM BREMSUNG )  EINES REGALFOERDERZEUGES
AUF DIE FOERDERLEISTUNG BEI CHAOTISCH
ORGANISIERTEM ZUGRIFF.
KRANSPIELART: EINZELSPIELE, NUR AUSLAGERUNG

30 REM  PROGRAMMABLAUF
40 REM  AEUSSERE SCHLEIFE (2 DURCHLAEUFE):
     1.DURCHLAUF:SIMULATIONSDATEN 1 EINLESEN
     2.DURCHLAUF:SIMULATIONSDATEN 2 EINLESEN

50 REM INNERE SCHLEIFE (200-MAL)
        FACHADRESSE AUSWAEHLEN
      1.FAHRDAUER ZUM FACH
       2.GABELSPIEL AUFNEHMEN
      3.FAHRDAUER ZUM ABGEBEPUNKT
      4.GABELSPIEL ABSETZEN
        GESAMTDAUER (1 BIS 4) AUFADDIEREN

60 REM AUESSERE SCHLEIFE: GESAMTDAUER ANZEIGEN
                  AKKU GESAMTDAUER LOESCHEN
**********



90 DEFINT I,J,S:RANDOM
100 REM BEGINN AUESSERE SCHLEIFE
110 GOSUB 5000: REM EINLESEN FUER 1.DURCHLAUF
120 GOSUB 6000: REM SIMULATION 1. DURCHLAUF
125 PRINT"1.DURCHLAUF, GESAMTDAUER=";D;" SEKUN
DEN": D=0

130 GOSUB 7000:REM MODIFIZIEREN F. 2.DURCHLAUF
140 GOSUB 6000:REM  SIMULATION 2. DURCHLAUF
145 PRINT"2.DURCHLAUF, GESAMTDAUER=";D;" SEKUN
DEN
150 END:
REM
*********
```

```
1970 REM          MODUL FACHADRESSE AUSWAEHLEN

        ALLE WERTE INTEGER
    AUSGANGSWERTE: H=HORIZONTALE FACHNR.
                   V=VERTIKALE   FACHNR.

1972 REM
    VARIABLE:      I=LAUFVARIABLE

1974 REM

   EINGANGSWERTE: HN=MINIMALE HOR.FACHNR.
                  VN=MINIMALE VERT.FACHNR.
                  HX=MAXIMALE HOR.FACHNR.
                  VX=MAXIMALE VERT.FACHNR.

1976 REM             SP=ANZAHL GESPERRTE FAECHER
                  SP(SP+SP)=ADRESSEN GESPERRTER
                  FAECHER HOR./VERT.

2000 H=RND(1+HX-HN)+HN-1:V=RND(1+VX-VN)+VN-1

2010 FOR I=1 TO SP+SP STEP 2:IF H=SP(I) AND V=
P(I+1) THEN H=-1:V=-1
2020 NEXT I

2030 IF H=-1 AND V=-1 THEN 2000 ELSE RETURN:
REM**********


2970 REM      MODUL FAHRDAUER ERRECHNEN AUS WEG
              ---------------------------------

         ALLE WERTE: REAL


2972 REM
EINGANGSWERTE:
         V=FAHRGESCHWINDIGKEIT     M/S
         A=ANFAHRBESCHLEUNIGUNG    M/(S*S)
         B=BREMSVERZOEGERUNG       M/(S*S)
         W=WEG                     M
```

```
2974 REM            FORMELN, WENN DER WEG SO LANG IST,
                    DASS DIE VOLLE FAHRGESCHWINDIG=
                    KEIT ERREICHT WIRD:
2976 REM
W1=U*U/(A+A)        ANFAHRWEG
W3=U*U/(B+B)        BREMSWEG
W2=W-W1-W3          WEG VOLLFAHRT

T1=(W1+W1)/U        ANFAHRDAUER
T2=W2/U             DAUER VOLLFAHRT
T3=(W3+W3)/U        BREMSDAUER
2978 REM
GESAMTDAUER:
T=(W1+W+W3)/U       W2 IST IN GLEICHUNG ERSETZT.


2980 REM            FORMELN, WENN DER WEG SO KURZ
                    IST, DAS DIE VOLLE GESHWINDIG=
                    KEIT NICHT  ODER GERADE EBEN
                    ERREICHT WERDEN KANN:

2982 REM
W1+W3=W             ANFAHRWEG+BREMSWEG=GESAMTWEG
A/B=W3/W1           WEGE SIND UMGEKEHRT PROPORTIO=
                    NAL DEN BESCHLEUNIGUNGEN

        DARAUS ABGELEITET:
W1=W/(A/B+1)        ANFAHRWEG
W3=W-W1             BREMSWEG
2984 REM
T1=SQR((W1+W1)/A)           ANFAHRDAUER
T3=SQR((W3+W3)/B)           BREMSDAUER



3000 W1=U*U/(A+A):W3=U*U/(B+B):
IF W1+W3<W
        THEN T=(W1+W+W3)/U
        ELSE W1=W/(A/B+1):W3=W-W1:T=SQR((W1+W1
)/A)+SQR((W3+W3)/B)
3010 RETURN:
REM**********
```

```
3970 REM          MODUL FAHRDAUER VON/NACH FACH=
                  ADRESSE ERMITTELN
                  ---------------------------------

3972 REM
EINGANGSWERTE INTEGER
H =  HOR.  FACHNR. NACH-FACH
V =  VERT.        -"-
G =  HOR.  FACHNR. VON-FACH
X =  VERT.        -"-
3974 REM
EINGANGSWERTE REAL
TH= HORIZONTALE FACHTEILUNG M
TV= VERTIKALE    -"-
3976 REM
AUSGANGSWERTE REAL
T = MASSGEBENDE FAHRZEIT  S
VARIABLE REAL
T1=ZWISCHENSPEICHER FAHRZEIT



3978 REM          AUSSERDEM: ALLE BEIM MODUL
                  FAHRDAUER (ZEILE 3000)
                  BENOETIGTE WERTE


4000 W=ABS(H-G)*TH:A=AH:B=BH:V=VH:GOSUB 3000:
T1=T:REM FAHRZEIT HOR.
4010 W=ABS(X-V)*TV:A=AV:B=BV:V=VV:GOSUB 3000:
        IF T1>T THEN T=T1
4020 RETURN:
REM**********


4970 REM          DATENMODUL
                  ----------
        DIES MODUL LIEFERT DIE DATEN FUER DEN
        1.SIMULATIONSLAUF


5000 DATA 3,     0,0,     0,1,     0,2     :REM GE
SPERRTE FAECHER
```

```
SP=ANZAHL GESPERRTER FAECHER
SP(1,2)=EINLAGERUEBERGABE
SP(3,4)=AUSLAGERUEBERGABE
SP(5,6)=FEUERLOESCHPALETTE
5010 READ SP:FOR I=1 TO SP+SP:READ SP(I):NEXT

5020 DATA 0,42, 0,12,   1.35,   2.2      :REM F
ACHTEILUNG
HN,HX=KLEINSTE/GROESSTE FACHNR.HOR.
VN,VX=KLEINSTE/GROESSTE FACHNR.VERT.
TH,TX=FACHTEILUNG HOR.,VERT.
5030 READ HN,HX,VN,VX,  TH,      TV

5040 DATA .5,  1,   .5,   1,   2,    1     :REM
AV,VV,BV=ANFAHRBESCHL.,GESCHW.,BREMSVERZ. VER
TIKAL
AH,VH,BH= DITTO                      HORIZONTAL
5050 READ AV,  VV,   BV,   AH,   VH,   BH

5060 DATA       20,     24    :REM GABELSPIELDAUER
GA=AUFNAHME
GB=ABSETZEN
5070 READ        GA,     GB

5080 RETURN:
REM**********
```

```
5970 REM MODUL SIMULATION, 200 KRANSPIELE
     ---------------------------------
6000 FOR S=1 TO 200:GOSUB 2000: REM FACHADRE
SSE AUSWAEHLEN
6010 GOSUB 4000:      REM FAHRDAUER ERMITTELN
6020 D=D+T+GA+T+GB:   REM ZEIT KUMULIEREN
6030 PRINT@960,S;" SPIELE.  ";:NEXT S
6040 RETURN:
REM**********
```

```
6970 REM          MODUL DATEN FUER 2.SIMULATIONS
LAUF MODIFIZIEREN
                  ------------------------------
-----------------
7000 AV=.4:BV=.4
7010 AH=.8:BH=.8
7020 RETURN
```

11. Lektion

11. LEKTION

Betrieb mit Disketten

Die Programmlistings dieser Lektion sind als Anhang beigefügt.

Eine vollständige Behandlung eines Diskettenbetriebssystemes erfordert den gleichen Zeitaufwand wie ein BASIC-Kursus.

Ziel dieser Lektion kann nur sein:

- Vorstellungen über ein Diskettenbetriebssystem zu bekommen.
- Begriffe zu erlernen, um Fachartikel und Fachbücher besser zu verstehen.

Wir behandeln als Beispiel eines Diskettenbetriebssystemes das Betriebssystem "NEWDOS" für den TRS-80.
Es gibt für verschiedene Computer unterschiedliche Betriebssysteme.
Als systemübergreifendes Betriebssystem ist das Betriebssystem CP/M eingeführt.

> Es sei mit aller Deutlichkeit darauf hingewiesen, daß die Tatsache, daß bei einem käuflichen Programm die Eigenschaft "CP/M-kompatibel" angegeben sit, nicht hinreichend ist, um sicherzustellen, daß dies Programm auf jedem Computer unter dem Betriebssystem CP/M ablauffähig ist.

Beim Betrieb mit Disketten kommt zu den in der ersten Lektion ge-

nannten Betriebsarten eine weitere Betriebsart hinzu, das

Diskettenbetriebssystem (DOS)

Im folgenden betrachten wir als Beispiel für ein DOS das Diskettenbetriebssystem "NEWDOS Version 2.1.", welches seit Anfang 1979 für den TRS-80 angeboten wird.

Organisation der Diskette:

Die korrekte Bezeichnung für eine Diskette, wie sie für den TRS-80 oder den Video Genie benötigt wird, ist:
Minidiskette, single sided, single density, soft sector.

Der Einfachheit halber sprechen wir – obwohl nicht korrekt -- von "Diskette". Wir meinen aber immer die o. g. Minidiskette.

Eine fabrikneue Diskette muß, bevor man sie als Speichermedium verwenden kann, initialisiert werden. Bei diesem Vorgang werden die Spuren der Diskette erstmalig beschrieben. Bei dieser Gelegenheit werden sie auf fehlerfreie Aufzeichnung überprüft, als fehlerhaft erkannte Spuren werden als nicht verwendbar gekennzeichent.

NEWDOS ermöglicht die Initialisierung mit wahlweise einem der folgenden Programme:

COPY/CMD	kopiert eine komplette Diskette
BACKUP/CMD	wie COPY/CMD, die Directory wird jedoch neu organisiert.
FORMAT/CMD	initialisiert eine Datendiskette.

Directory:
Die Datei DIR/SYS enthält ein Verzeichnis über die Belegung der Spuren. Jede Datei hat einen Namen, der in DIR/SYS eingetragen ist. DIR/SYS ist somit der Schlüssel zur Diskette, von seiner Bedeutung her in etwa vergleichbar mit der Klingelknopfleiste am Eingang eines Hochhauses mit vielen Wohnungen: Wenn die Datei DIR/SYS nicht mehr lesbar ist, ist ein Zugriff zu den übrigen Dateien der Diskette praktisch nicht mehr möglich.

Daten- oder Systemdiskette:
Die Diskette in Laufwerk 0 muß das DOS enthalten. Eine Diskette, die das DOS enthält, hat den Namen "Systemdiskette". Disketten, die kein DOS enthalten, haben den Namen "Datendiskette".

Insgesamt ergeben sich folgende Zuordnungen:

Laufwerk 0	muß Systemdiskette enthalten.
Laufwerk 1 bis 3	kann System- oder Datendisketten enthalten.

Eine Diskette hat normalerweise eine Kapazität von 35 Spuren zu je 2 Granules:

1 Diskette	35 Spuren	70 Granules
1 Spur	2 Granules	10 Sektoren
1 Granule	5 Sektoren	
1 Sektor	255 Bytes	

Für Anwendungen nutzbar:

bei Systemdiskette ca. 50 Granules = 63750 Bytes
bei Datendiskette 67 Granules = 85425 Bytes

Dateien werden granulesweise angelegt: die kürzeste mögliche Datei ist somit 1 Granule lang.

Zur Prüfung der für eine Diskette lebensnotwendigen Datei DIR/SYS gibt es ein Hilfsprogramm DIRCHECK/CMD, welches die DIR/SYS überprüft. Als Anlage 1 ist ein Ausdruck dieses Programmes mit einer Systemdiskette enthalten.

BASIC/CMD

Dieses Modul erweitert das ROM-BASIC um einige zum Diskettenbetrieb erforderliche BASIC-Befehle. Gleichzeitig wird der freie RAM-Speicher leider reduziert, denn die Erweiterung benötigt Speicherplatz. Als "Faustregel" kann gelten, daß bei einem System mit 48 k-Byte RAM dem Anwender für seine BASIC-Programme lediglich 38 k-Byte RAM zur freien Verfügung steht.

Die eben gemachte Aussage läßt sich verallgemeinern in folgende Aus-

sage:

Bei einem Computersystem mit 64 k-Byte Gesamtspeicher (RAM und ROM) belegt das DOS und der BASIC-Interpreter etwa 25 k-Byte, so daß dem Anwender ein nutzbarer Speicherraum von etwa 39 k-Byte zur Verfügung steht.

An dieser Stelle sei ein weiterer Hinweis erlaubt:
Für den Homecomputer APPLE II wird eine 16-k RAM-Zusatzkarte angeboten. Der unbefangene Homecomputeranwender könnte vermuten, daß diese Karte die Speicherkapazität seines Homecomputers unter BASIC um 16 k-Byte vergrößert. Das ist nicht der Fall, diese Karte vergrößert die Speicherkapazität des APPLE II-Computers unter dem normalen APPLESOFT-BASIC nicht. Die Karte soll lediglich den 16k-Speicherverlust ausgleichen, der durch die Verwendung einer anderen Karte (diese Karte enthält eine Z80-Zentraleinheit) hervorgerufen wird.

Aus den eben gemachten Ausführungen kann man auch schliessen, daß ein Betrieb des TRS-80 oder des Video Genies mit 16 k-Byte RAM und DOS dem Benutzer nur ca. 7 k-Byte netto für seine Anwendungen läßt.

Sofern beim NEWDOS genügend Freiraum im Speicher vorhanden ist (im Klartext: sofern die Ausbaustufe 32 oder 48 k-Byte RAM ist), lassen sich die Befehle der Betriebsart DOS als BASIC-Befehle codieren.

Als Beispiel für eine solche Nutzung ist als Anlage 2 ein Kaltstart-Hilfsprogramm beigefügt.
Die dort enthaltenen Befehle CMD"xxx" sind Befehle aus der DOS-Betriebsart.
Das Programm der Anlage 2 ist ein Anpassprogramm, das nach Einschalten des Computersystems ausgeführt wird. Es initialisiert den String TIME$. Dieser String enthält die Uhrzeit. Mit dem Modul KURZ1 wird der Drucker- und Tastatur-DCB modifiziert.

BASIC/CMD erleichtert das Editieren von Programmen durch eine Funktion "Blättern", die mit den Tasten Pfeil auf und Pfeil ab bedient wird. Weiterhin kann man die Kommandos

EDIT als E und
LIST als L abgekürzt schreiben.

Als neue Funktionen kommen hinzu:

RENUM zum neuen Durchnummerieren eines Programmes
REF zum Erstellen einer Kreuzreferenz (vergl. Anlage 2).

RENUM und REF haben verschiedene Parameter zur optimalen Einsatzanpassung.

MERGE dient zur Verkettung von Programmodulen unter Steuerung durch die Zeilennummer.

Bei der Erweiterung des BASIC-Befehlsumfanges fallen besonders auf (vergl. dazu Anlage 3, das Programm enthält Anwendungsbeispiele für die genannten Befehle):

DEF FN ist eine besondere Art des Unterprogrammaufrufes mit lokalen Symbolen. Das Fehlen dieses BASIC-Befehls im Grund-BASIC-Dialekt des Video Genie oder TRS-80 ist ein wesentlicher Mangel.

INSTR ermöglicht das schnelle Auffinden eines kleinen Strings in einem größeren String. Für Applikationen mit Textbearbeitung ist dieser Befehl ein "Muß".

LINEINPUT ermöglicht die Eingabe von Texten, die alle Satzzeichen enthalten können. Für Applikationen mit Textbearbeitung ist dieser Befehl ein "Muß".

MID$ = . . . Es ist möglich, einen Teil eines Strings zu verändern. Für Applikationen mit Textverarbeitung ist dieser Befehl ein "Muß". Beachten Sie, daß MID$ hier links vom Zuweisungszeichen steht!

Dateizugriffe laufen über die Data Control Blocks (DCB – Sie erinnern, daß auch die Device Control Blocks mit DCB abgekürzt werden, obwohl es sich hier um unterschiedliche Dinge handelt !).

Der BASIC-Dialekt des TRS-80 und des Video Genies hat einen eigenen Befehlssatz für die Dateiverarbeitung. Es wird unterschieden zwischen

sequentiellem und direktem Zugriff:

sequentiell	direkt
OPEN#	OPEN#
LINEINPUT#	READ#
INPUT#	
PRINT#	PUT#
CLOSE#	CLOSE#

Der direkte Zugriff ermöglicht das unmittelbare Ausfinden einer Information innerhalb einer Datei anhand eines "Schlüssels". Das Thema "direkter Zugriff" würde den Rahmen dieses Kursus sprengen, deshalb nur einige Hinweise:

- bei einer Kudnenstammdatei ist die "Kundennummer" so gut wie nie der Schlüssel für einen direkten Zugriff.
- das Entwerfen von Applikationen mit Direktzugriffsdateien ist eine relativ komplizierte Angelegenheit die umfassende Kenntnisse erfordert. Mit anderen Worten: dies ist eine Aufgabe für EDV-Fachleute, dies ist keine Aufgabe für nicht-EDV-Fachleute.

Beim seqentiellen Zugriff wird eine Datei von vorn beginnend durchgelesen (LINEINPUT#-Befehl) oder von vorn beginnend ausgegeben (PRINT#-Befehl).
Es wird kein Schlüssel verwendet, die Diskette stellt nichts anderes dar als eine Magnetbandkassette mit schnellerem Zugriff.

Eine Datei muß vor ihrer Verarbeitung "eröffnet" werden. Hierbei wird der Datei ein DCB zugewiesen und die Datei wird auf den sich im Zugriff befindlichen Disketten in den Dateien DIR/SYS dieser Disketten gesucht. Nach Eröffnen erfolgt die Verarbeitung der Datei. Abschliessend muß ein CLOSE#-Befehl durchgeführt werden. Der CLOSE#-Befehl gibt den DCB frei und verändert – sofern PRINT#-Befehle ausgeführt wurden – die Datei DIR/SYS.

Sofern PRINT#-Befehle ausgeführt wurden, ein CLOSE#-Befehl jedoch nicht ausgeführt wurde, stimmt die Datei DIR/SYS nicht mehr mit dem Inhalt der Diskette überein. Um den Verlust von Daten dieser Diskette zu vermeiden, muß man die fehlerhaft abgeschlossene Datei mit dem Kommando KILL in der Datei DIR/SYS entfernen.

Beim Testen führt NEWDOS von sich aus selbstständig CLOSE-Befehle durch. Diese Automatisierung erleichtert das Testen wesentlich.

Dateinamen

Die Dateinamen enthalten einen Suffix. Die Empfehlungen für den Suffix lauten:

/CMD	ausführbare Maschinencodemodule
/CIM	nicht ausführbare Maschinencodemodule
/TXT	Textdateien (ASCII-Code)
/DAT	Dateien (nicht ausschliesslich ASCII-Code)
/SOU	Assembler-Quellenprogramme
/BAS	BASIC-Interpreterprogramme
/SYS	Diskettenbetriebssystembestandteile

Weiterhin enthält das DOS einen Zugriffsschutz, der jedoch den folgenden Mangel hat:

– In der Bedienungsanleitung für NEWDOS steht, wie er umgangen werden kann.

Datensicherung

Man unterscheidet die Sicherung ganzer Disketten und die Sicherung einzelner Dateien.

Die Sicherung kann erfolgen:

ganze Disketten:	BACKUP/CMD oder COPY/CMD
Dateien:	
jeder Datentyp	COPY/CMD
/BAS	LOAD/SAVE (BASIC)
/SOU	EDTASM/CMD

Eine Sicherung der SYS-Dateien ist nur im Rahmen der Sicherung ganzer Disketten möglich, da diese Dateien zum Teil nicht über DIR/SYS vollständig verwaltet werden und deshalb bestimmte Sektoren belegen müssen.

Für Systemänderungen, bei denen Inhalte der /SYS-Dateien geändert

werden müssen, gibt es das Hilfsprogramm SUPERZAP/BAS. SUPERZAP/BAS ermöglicht Zugriff zu Dateien unter Vermeidung der DIR/SYS und das "Reparieren" von Sektoren mit Lesefehlern (Parity Error), wobei jedoch nach Bereinigung des Lesefehlers die betroffene Datei einen fehlerhaften Inhalt hat. Der Sektor ist nach "Reparatur" wiederverwendbar.

Grundsätzlich muß jede Systemdiskette mehrmals vorhanden sein:

1 Mutterdiskette mit dem Originalsystem
je 1 angepasste Systemdiskette
je 1 Kopie 1 der angepassten Systemdiskette (je Mitarbeiter)
je 1 Arbeitssystemdiskette (je Mitarbeiter)

Somit hat jeder Mitarbeiter seine eigene Systemdiskette und eine Kopie davon.
Die ersten zwei Systemdiskettenarten werden beim Systembetreuer unter Verschluß aufgehoben. Bei Problemen mit den Diskettenlaufwerken sind dadurch die Mitarbeiter gezwungen, den Systembetreuer zur Erstellung von Kopien hinzuzuziehen.

Die Dateien von Datendisketten und normale Programme gelten als gesichert, wenn sie auf mindestens zwei unterschiedlichen Disketten vorhanden sind.

Auftreten von Lesefehlern

Beim Auftreten eines Lesefehlers auf einer Diskette ist davon auszugehen, daß das Computersystem defekt sein kann.

Folgende Vorgehensweise ist angebracht:

- die laufende Arbeit wird unterbrochen
- mit SUPERZAP/BAS wird der fehlerhafte Sektor festgestellt
- der fehlerhafte Sektor wird mit SUPERZAP/BAS erneut verwendbar gemacht
- die Datei wird unter dem gleichen Namen von der Sicherungsdiskette her neu auf die Diskette aufgespielt, die den Fehler hatte
- die unterbrochene Arbeit wird neu aufgesetzt bzw. fortgesetzt.

In folgenden Fällen ist eine Korrektur mit SUPERZAP/BAS praktisch

nicht möglich:

Fehler in der Directory (DIR/SYS)
Fehler "Sector not found"

In diesen Fällen muß die Diskette neu initialisiert werden. Sämtliche auf ihr befindliche Dateien müssen neu aufgespielt werden, bevor die Arbeit fortgesetzt werden kann.

Beide eben beschriebenen Vorgehensweisen belassen die Diskette, bei welcher der Fehler auftrat, im Computersystem. Sofern das Computersystem tatsächlich defekt ist, werden auf diese Weise keine weiteren Disketten in Mitleidenschaft bezogen.

NOTIZEN

Anlage

ANLAGE 1 zur 11. Lektion

Beispiel für DIRECHECK/CMD

```
NEWDOS      00 00/

BASIC/CMD      I       EOF = 19/119     2 EXTS   20 SECTORS
BOOT/SYS       SIP=6   EOF = 5/0       1 EXTS   5 SECTORS
COPY/CMD       IP=6    EOF = 4/253      1 EXTS   5 SECTORS
DCU/CMD                EOF = 4/0        1 EXTS   5 SECTORS
DIR/SYS        SIP=5   EOF = 10/0       1 EXTS   10 SECTORS
DIRCHECK/CMD           EOF = 12/236     1 EXTS   15 SECTORS
DISASSEM/CMD           EOF = 19/134     1 EXTS   20 SECTORS
DISKDUMP/BAS           EOF = 9/242      1 EXTS   10 SECTORS
EDTASM/CMD             EOF = 31/154     2 EXTS   35 SECTORS
FORMAT/CMD     IP=6    EOF = 14/8       1 EXTS   15 SECTORS
LA34                   EOF = 1/109      1 EXTS   5 SECTORS
LEVEL1/CMD             EOF = 19/0       1 EXTS   20 SECTORS
LMOFFSET/CMD           EOF = 6/224      1 EXTS   10 SECTORS
LV1DSKSL/CMD           EOF = 2/141      1 EXTS   5 SECTORS
RTY80                  EOF = 9/0        1 EXTS   10 SECTORS
SUPERZAP               EOF = 53/106     3 EXTS   55 SECTORS
SUPERZAP/COM           EOF = 25/189     2 EXTS   30 SECTORS
SYS0/SYS       SIP=7   EOF = 12/93      1 EXTS   15 SECTORS
SYS1/SYS       SIP=7   EOF = 4/142      1 EXTS   5 SECTORS
SYS11/SYS      SIP=7   EOF = 4/236      1 EXTS   5 SECTORS
SYS12/SYS      SIP=7   EOF = 4/164      1 EXTS   5 SECTORS
SYS13/SYS      SIP=7   EOF = 3/9        1 EXTS   5 SECTORS
SYS2/SYS       SIP=7   EOF = 4/52       1 EXTS   5 SECTORS
SYS3/SYS       SIP=7   EOF = 4/76       1 EXTS   5 SECTORS
SYS4/SYS       SIP=7   EOF = 4/186      1 EXTS   5 SECTORS
SYS5/SYS       SIP=7   EOF = 4/203      1 EXTS   5 SECTORS
SYS6/SYS       SIP=7   EOF = 13/33      1 EXTS   15 SECTORS

1 FREE GRANULES.      0 LOCKED-OUT GRANULES.

NEWDOS     DIRECTORY CHECK & LIST COMPLETED.
```

ANLAGE 2 zur 11. Lektion

Beispiel für ein Kaltstart-Hilfsprogramm

```
10 DEFINT A-Z:I=INP(233)
11 CLS:PRINT@800,"":FORI=0TO255:POKE 15616+I,I:NEXT
20 FORI=1TO 500:NEXT:I=INP(233)
21 A=15415:IFPEEK(A)=58ANDPEEK(A+3)=58THEN380:REM WENN UHR EIN
26 OUT254,1:PRINT"

RAM-TEST
30 FORA=-1TO-255STEP-1:POKEA,ABS(A):POKEA-&H4000,ABS(A):NEXT
32 FOR A=-1 TO -255 STEP -1:IF PEEK(A)<>ABS(A) THEN PRINT"ERROR @";A
34 IFPEEK(A-&H4000)<>ABS(A)THENPRINT"ERROR @";A-&H4000
60 NEXT:OUT254,0:CMD"CLOCK":PRINT"KURZSCHRIFT (J/N)":A$=INSTR$(1):PRINT:IF A$="J
" OR A$="J"THEN CMD"KURZ1":GOTO 110
65 PRINT"KLEINSCHRIFT (J/N)":A$=INSTR$(1):PRINT:IF A$="J" OR A$="J" THEN CMD "KL
EIN
70 PRINT"UHR STELLEN ? (J/N)
100 A$=INSTR$(1):IF A$="N" OR A$="n"THEN 180
110 INPUT"STUNDE = (0 BIS 23)";H
120 IF H<0 OR H>23 PRINT"PFUI !":GOTO 110
130 INPUT"MINUTE = (0 BIS 59)";M
140 IF M<0 OR M>59 PRINT"PFUI !":GOTO 130
150 POKE &H4043,H:POKE &H4042,M:POKE &H4041,30
160 PRINT:GOTO 70

180 PRINT"NEUES DATUM ? (J/N)
210 A$=INSTR$(1):IF A$="N"OR A$="n"THEN 290
220 INPUT"TAG = (1 BIS 31)";T
230 IF T<1 OR T>31 PRINT"PFUI !":GOTO 220
240 INPUT"MONAT = (1 BIS 12)";M
250 IF M<1 OR M>12 PRINT"PFUI !":GOTO 240
260 INPUT"JAHR = (80 BIS 90)";J
270 IF J<80 OR J>90 PRINT"PFUI !";:GOTO 260
280 POKE &H4044,J:POKE &H4045,T:POKE &H4046,M
290 PRINT"DATUM UND UHRZEIT:
TT.MM.JJ HH:MM
310 B$=TIME$
320 PRINTMID$(B$,4,2)+"."+LEFT$(B$,2)+"."+MID$(B$,7,8)
340 PRINT"STIMMT ES SO ? (J/N)
350 A$=INSTR$(1):IF A$="J" OR A$="J" PRINT:GOTO 380 ELSE 70

380 CMD"DIR
```

```
    0  11 60 120 140
    1  20 26 30/2 32/2 60 65 100 210 230 250 350
    2  320/2
    3  21
    4  320
    7  320
    8  320
   12  250
   23  120
   30  150
   31  230
   58  21/2
   59  140
   70  160 350
   80  270
   90  270
  110  60 120
  130  140
  180  100
  220  230
  233  10 20
  240  250
  254  26 60
  255  11 30 32
  260  270
  290  210
  380  21 350
  500  20
  800  11
15415  21
15616  11
   A   10 21/3 30/5 32/4 34/3 60/$3 65/$3 100/$3 210/$3
       350/$3
   B   310/$ 320/$3
   H   110 120/2 150
   I   10 11/3 20/2
   J   260 270/2 280
   M   130 140/2 150 240 250/2 280
   T   220 230/2 280
   Z   10
```

ANLAGE 3 zur 11. Lektion

Beispiel für ein Adressenverwaltungsprogramm

```
10 CLEAR 15000:DEFINT A-Z
20 REM ADRESSDATEI, R.-F.MATTHAEI, 7.8.81
DER LISTENAUFBAU IST ALPHANUMERISCH SORTIERT VERKETTET
30 DEF FNR$(A$)= LEFT$(A$,LEN(A$)-1):'RECHTES ZEICHEN ENTFERNEN
40 DEF FNL$(A$)=RIGHT$(A$,LEN(A$)-1):'LINKES ZEICHEN ENTFERNEN
50 'ENTFERNT DAS A. ZEICHEN AUS DEM STRING A$ UND FUEGT RECHTS
EIN SPACE AN:
60 DEF FNK$(A$,A)=LEFT$(A$,A-1)+MID$(A$,A+1,LEN(A$)-A)+" "
70 DIM P(500),D$(500),D1$(500),D2$(500),D3$(500)
80 GOSUB 1230:'INIT

90 INPUT"STANDARDDATEINAME (OHNE EXTENSION):";DA$
100 IF DA$="" OR ASC(DA$)<ASC("A") THEN PRINT"PFUI !":GOTO 90
110 PRINT"MENUE
----  (GGF. ,,HILFE'' EINGEBEN)
120 FOR EI=0 TO 1 STEP 0
130 PRINT"HOECHSTER POINTER=";Q
140 PRINT"NAME....";:GOSUB 1620:PRINT
150 IF LEFT$("HILFE",LEN(D$))=D$ THEN 720
160 IF LEFT$("LISTE",LEN(D$))=D$ THEN 520
170 IF LEFT$("LOESCHE",LEN(D$))=D$ GOSUB 620:NEXT EI
180 IF LEFT$("SAVE",LEN(D$))=D$ THEN 760
190 IF LEFT$("LOAD",LEN(D$))=D$ THEN 960
200 IF LEFT$("KILL",LEN(D$))=D$ THEN 1510
210 IF LEFT$("FREE",LEN(D$))=D$ THEN CMD"FREE":NEXT EI
220 IF LEFT$("DIR",LEN(D$))=D$ THEN CMD"DIR":NEXT EI
230 IF LEFT$("END",LEN(D$))=D$ THEN 1130
240 IF LEFT$("DATEI",LEN(D$))=D$ THEN 1460
250 IF LEFT$("INIT",LEN(D$))=D$ GOSUB 1230:NEXT EI
260 IF LEFT$("SUCHE",LEN(D$))=D$ THEN 1320
270 IFLEFT$("SPACE",LEN(D$))=D$ THEN PRINT"FREIER STRINGSPACE=
";FRE(A$);" BYTES":NEXT EI

280 D1$="":LINEINPUT"ANREDE......";D1$
290 '
```

```
300 IF D1$=""THEN NEXT EI:'LEERSTRING=ABBRUCH DER FUNKTION
310 LINEINPUT"STRASSE.....";D2$
320 LINEINPUT"ORT.........";D3$
330 GOSUB 340:NEXT EI

340 D$(Q+1)=D$
350 D1$(Q+1)=D1$
360 D2$(Q+1)=D2$
370 D3$(Q+1)=D3$
380 IF Q=0 THEN 490
390 IF Q=500 THEN 520:'MAX.500 GLIEDER
400 IF LEFT$(D$,1)<"A" OR LEFT$(D$,1)>"Z" THEN KH=P(0):GOTO 420
410 KH=P(ASC(D$)-ASC("A"))
420 K2=0
430 FOR I=0 TO 1 STEP 0
440 IF D$(Q+1)<D$(KH) THEN KH=K2:GOTO 500
450 IF D$(Q+1)=D$(KH) THEN PRINT D$(KH);" SCHON VORHANDEN !
":RETURN
460 K2=KH:KH=P(KH)
470 IF KH=0 THEN 490
480 NEXT I

490 P(K2)=Q+1:P(Q+1)=0:Q=Q+1:GOTO 510
500 P(Q+1)=P(KH):P(KH)=Q+1:Q=Q+1
510 RETURN

520 PRINT"
BEENDEN MIT X - TASTE
DATEI="DA$
530 INPUT"
AB BUCHSTABEN ";D$
540 IF D$<"A" OR D$>"Z" OR LEN(D$)<>1 THEN PRINT"PFUI !":GOTO 530
550 KH=P(ASC(D$)-ASC("A"))
560 FOR I=1 TO 2 STEP 0:PRINT D$(KH)
570 PRINTD1$(KH):PRINTD2$(KH):PRINTD3$(KH)
580 D$="":D$=INKEY$:IF D$="X" THEN 610
590 IF P(KH)=0 THEN 610
600 KH=P(KH):NEXT I
610 PRINT:PRINT:NEXT EI

620 PRINT"
LOESCHEN.

WAS LOESCHEN :
630 GOSUB 1620
640 KH=P(0):IF D$=D$(KH) THEN 700
650 FOR I=0 TO 1 STEP 0
660 IF D$=D$(KH) THEN 690
670 K2=KH:KH=P(KH):IF KH=0 THEN PRINT"NICHT GEFUNDEN
":RETURN

680 NEXT I
690 P(K2)=P(KH):GOTO 710
700 P(0)=P(KH)
710 PRINTD$;" geloescht !
":D$(KH)="":D1$(KH)="":RETURN
730 PRINT"
END     = PROGRAMMLAUF BEENDEN
```

```
DATEI    = STANDARDDATEINAME
INIT     = NEUAUFBAU DER DATEI
SUCHE    = SUCHEN EINES NAMENS
SPACE    = ANZEIGE FREIER STRINGSPACE
"
740 NEXT EI
750 '
760 PRINT"SPEICHERN AUF FLOPPY":TN$=""
770 INPUT"DATEINAME (OHNE EXTENSION) ODER NUR ENTER";TN$
780 IF TN$="" THEN TN$=DA$
790 TN$=TND$+"/TXT,A"
800 'EINSTIEGE MIT "A" BIS "Z" LOESCHEN
810 FOR S=ASC("A") TO ASC("Z")
820 D$=CHR$(S):GOSUB 640
830 NEXT S
840 OPEN"O",1,TN$
850 KH=P(0)
860 FOR I=0 TO 1 STEP 0
870 IF D$(KH)="" AND D1$(KH)="" THEN 920
880 PRINT#1,D$(KH):PRINT D$(KH)
890 PRINT#1,D1$(KH):PRINT D1$(KH)
900 PRINT#1,D2$(KH):PRINTD2$(KH)
910 PRINT#1,D3$(KH):PRINTD3$(KH)
920 IF P(KH)=0 THEN 940
930 KH=P(KH):NEXT I
940 CLOSE 1:PRINT:PRINT:NEXT EI
950 '
960 PRINT"VON FLOPPY LADEN":D$=""
970 INPUT"DATEINAME (OHNE EXTENSION) ODER NUR ENTER";D$
980 IF D$="" THEN D$=DA$
990 D$=D$+"/TXT"
1000 OPEN"I",1,D$
1010 IF EOF(1) THEN PRINT"DATEI GIBT ES NICHT / IST LEER":NEXT EI

1030 GOSUB 1230:'INIT
1040 FOR I=0 TO 1 STEP 0
1050 LINEINPUT#1,D$:PRINT D$
1060 LINEINPUT#1,D1$:PRINT D1$
1070 LINEINPUT#1,D2$:PRINTD2$
1080 LINEINPUT#1,D3$:PRINTD3$
1090 GOSUB 340
1100 IF EOF(1) THEN PRINT:PRINT"DATEIENDE":CLOSE:NEXT EI
1110 NEXT I

1130 PRINT"PROGRAMM BEENDEN
HAST DU AUCH SAVE NICHT VERGESSEN ?
-----------------------------------
EINGABE: END ODER SAVE
1170 D$="":INPUT D$
1180 PRINT:IF D$="END" THEN PRINT"PROGRAMMENDE":END
1190 PRINT:IF D$="SAVE" THEN 760
1200 PRINT:NEXT EI

1220 'INITIALISIEREN DER POINTER
1230 FOR I=0 TO Q:P(I)=0:NEXT:'POINTER LOESCHEN
1240 FOR S=ASC("A")-ASC("A") TO ASC("Z")-ASC("A")
1250 D$(S+1)=CHR$(S+ASC("A"))
1260 P(S)=S+1
```

```
1270 NEXT S
1280 Q=S
1290 PRINT"FERTIG INITIALISIERT"
1300 RETURN

1320 PRINT"SUCHEN

WAS SUCHEN:
1340 GOSUB 1620
1350 IF LEFT$(D$,1)<"A" OR LEFT$(D$,1)>"Z" THEN KH=P(0):GOTO 1370
1360 KH=P(ASC(D$)-ASC("A"))
1370 FOR I=0 TO 1 STEP 0
1380 IF D$=D$(KH) THEN 1410
1390 K2=KH:KH=P(KH):IF KH=0 THEN PRINT"NICHT GEFUNDEN
":NEXT EI

1400 NEXT I
1410 PRINT D$
1420 PRINTD1$(KH)
1430 PRINTD2$(KH):PRINTD3$(KH)
1440 NEXT EI

1460 PRINT"STANDARDDATEINAME=";DA$:D$=""
1470 INPUT"NEUER STANDARDDATEINAME ODER NUR ENTER";D$
1480 IF D$<>"" THEN DA$=D$
1490 NEXT EI
1500 '
1510 INPUT"WELCHE DATEI SOLL GELOESCHT WERDEN (OHNE EXTENSION)";D$
1520 D$=D$+"/TXT"
1530 KILL D$
1540 NEXT EI

1560 'UPRO, ENTFERNT AUS D$ DAS ERSTE ZEICHEN VON IN$
1570 IN=INSTR(D$,IN$)
1580 IF IN<>0 THEN D$=FNK$(D$,IN):D$=FNR$(D$):GOTO 1570
1590 RETURN
1600 '
1610 'Stringinput von D$ mit Abschneiden der fuehrenden/
nachlaufenden spaces
1620 LINEINPUT D$
1630 IN$="  ":GOSUB 1570:'ENTFERNT SPACEKETTEN
1640 IF  LEFT$(D$,1)=" " THEN D$=FNL$(D$):GOTO 1640
1650 IF RIGHT$(D$,1)=" " THEN D$=FNR$(D$):GOTO 1650
1660 RETURN
1670 END
```

NOTIZEN

NOTIZEN

NOTIZEN

NOTIZEN

NOTIZEN

NOTIZEN

Auch Sie brauchen ELCOMP!

Preis pro Heft 29,80 DM
incl. MwSt. und Versand

Zurückliegende Hefte sind zu Originalpreisen noch verfügbar.

ELCOMP

HOFACKER-VERLAG
Ing. W. Hofacker GmbH
Tegernseer Straße 18
D-8150 Holzkirchen/Obb.

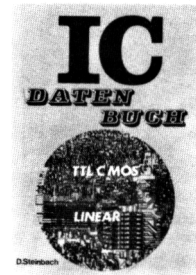
Die Fachzeitschrift für MICROCOMPUTER
Eine unentbehrliche Informationsquelle für alle Elektroniker

Microcomputer-Anwendungsbeispiele
Künstliche Intelligenz
Block-Strukturierte Programme
Datenverarbeitung im Kleinbetrieb
Club-Neuheiten
Computer und Kunst
Musik mit dem Computer
Monitore für 8080, 6800, 6502, Z 80, SC/MP, 2650, 1802
Eigenbau-Computersysteme
Interface-Techniken
Microcomputer KITs
Neue Produkte
Betriebssysteme für Floppys
Programmiertechniken
Software-Quellen
Programmierbeispiele
Soziale Aspekte der Microcomputertechnik
Technologische Neuheiten
Anwendungen in der Meß- und Regeltechnik
Anwendungen bei Funk-Amateuren

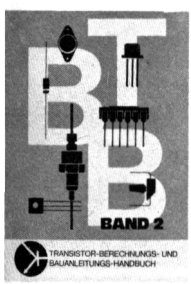
Hofacker-Bücher Deutsch

TBB-Transistor-, Berechnungs- und Bauleitungs-HB, Band 1, Hofacker
Völlig neu überarbeitete Auflage. Das Buch soll bei der täglichen Arbeit im Labor, in der Werkstatt oder am Elektronik-Hobbytisch Ihnen ein guter Begleiter sein. Berechnungsgrundlagen, Berechnungsbeispiele, Tabellen, Vergleichslisten, Digitaltechnik, Netzgeräte, BASIC-Programme zur Berechnung spezifischer Schaltungen usw. sind in übersichtlicher Form dargestellt. Ca. 300 Seiten.
Best.-Nr. 1 **29,80 DM**

Electronic im Auto, H. Gebauer
Mit Handbuch für Polizeiradar. Ein Buch für jeden technisch interessierten Autofahrer. Es zeigt Ihnen die vielen Möglichkeiten zur Verbesserung von Sicherheit, Leistung und Fahrkomfort in Ihrem Auto. Thyristorzündung, Beschleunigungsmesser, Drehzahlmesser, Batterieladegerät, Alarmanlagen u. v. a.
Best.-Nr. 3 **9,80 DM**



IC-Datenbuch, D. Steinbach
Daten- und Auswahllisten der gebräuchlichsten integrierten Schaltkreise. Digital und analog. Gerade bei ICs ist es wichtig die Anschlußfolgen genau zu kennen. Die wichtigsten TTL-Schaltkreise, NF-Verstärker, C-Mos Serie, lineare Schaltungen wie Operationsverstärker; Komparatoren, Spannungsregler, Trigger-Schaltungen, u. v. a. Das IC-Datenbuch wird auch Ihnen ein unentbehrlicher Begleiter bei allen Arbeiten mit integrierten Schaltungen sein.
Best.-Nr. 5 **9,80 DM**



TBB-Handbuch, Band 2, Hofacker
Dieses Buch ist die Fortsetzung des erfolgreichen Handbuches, Band 1. Ein Buch, das sich in der Hand des Praktikers bestens bewährt hat. Weitere neueste Schaltbeispiele und Berechnungsgrundl., Experimentier- und Versuchsbeschreibungen. Integrierte Spannungsregler, Wärmeableitung, Operationsverstärker Einführung, RC-Zeitglieder, Transistortester u. v. a.
Best.-Nr. 2 **19,80 DM**

IC-Handbuch, C. Lorenz
Ein Handbuch für digitale und lineare integrierte Schaltungen. Daten- und Auswahl-, Vergleichslisten, Gehäuseformen, Grundlagen, viele Schaltbeispiele, Printvorlagen, u. v. a. Alles über TTL-Technik, C MOS, MOS-Schaltungen, integrierte NF-Verstärker u. v. a.
Best.-Nr. 4 **19,80 DM**

IC-Schaltungen, D. Steinbach
Hier finden Sie eine gelungene Zusammenstellung der wichtigsten Anwendungsbeispiele aus dem Bereich der integrierten Schaltungen. TTL, C MOS, Linear. Alle Schaltungen sind übersichtlich und klar dargestellt und mit einer kurzen, jedoch sehr genauen Beschreibung versehen. Tastenenprellung, Zähler, Impulsgeber, Codierer, Dekodierer, Datenübertragung, Serien-Parallel-Wandler, Digitalvoltmeter u. v. a.
Best.-Nr. 6 **19,80 DM**

Elektronik Schaltungen, Hofacker
Die ideale Schaltungssammlung zum Basteln u. Experimentieren, Schaltungen mit Operationsverstärkern, Spannungsreglern, TTL, C-MOS-Schaltkreisen. MOS Uhr mit Weckerelektronischer Würfel u. v. a.
Best.-Nr. 7 **9,80 DM**

IC Bauanleitungs Handbuch -IC -KIT, C. Lorenz
Ein Bauanleitungsbuch mit vielen hochinteressanten Bauanleitungen aus dem Bereich der LSI Schaltungstechnik. Schaltbeispiele mit Printvorlagen zum Selbstherstellen der Leiterplatten mit genauesten Beschreibungen. Hochaktuell und brandneu: Funktionsgenerator XR 2206, MOS-Uhr mit Wecker, programmierbarem Wecktongenerator, Schlummerautomatik, IC-Netzteil, Experimentieranleitung und Grundkurs über Flip Flops, u. v. a. Zu allen Schaltungen finden Sie Platinenvorlagen oder Sie können die Experimentierschaltungen auf der Experimentierplatine WH-1 g durchführen. Über 100 Seiten.
Best.-Nr. 8 **19,80 DM**

Feldeffekttransistoren, C. Lorenz
Der Feldeffekttransistor (FET) gehört heute zu den interessantesten Bauteilen überhaupt. Wie man damit experimentiert, wie man seine Funktion versteht und wie man damit brauchbare u. hochinteressante Schaltungen aufbauen kann, zeigt Ihnen dieses Buch. Grundlagen, Kennlinienfelder, Tabellen, Rechenbeispiele, Anschlußbilder und eine Vergleichsliste für Feldeffektransistoren bilden den Kern dieser umfangreichen Darstellung. Alles in allem finden Sie hier eine praxisnahe und komplette Arbeitsunterlage, mit der Sie im Beruf und auch im Hobby erfolgreich arbeiten können.
Best.-Nr. 9 **9,80 DM**

Elektronik und Radio, C. Lorenz
4. Auflage. Völlig neu bearbeitet und stark erweitert. Eine Schritt für Schritt Einführung in die Radiotechnik mit vielen Bildern. Vom einfachen Diodenempfänger (Detektor) bis zu interessanten Sender- und Empfängerschaltungen (Minispione). IC-Radio, IC-Sender, Antennen, Berechnungsgrundlagen, Tabellen u. v. a. Über 150 Seiten.
Best.-Nr. 10 **19,80 DM**

NF-Verstärker, C. Lorenz
Grundlagen der integrierten NF-Verstärker, Berechnung von kompletten IC-NF-Verstärkerstufen. Anwendungsbeispiele mit den interessantesten und gebräuchlichsten Standard IC-NF-Verstärkern wie TBA 800, TBA 830, usw. Printvorlagen, Auswahltabellen, Experimentieranleitungen und Anschlußbilder machen dieses Buch zu einem unentbehrlichen Begleiter für alle, die sich m. NF-Verstärkern beschäftigen wollen.
Best.-Nr. 11 **9,80 DM**

BIS, Beispiele integrierter Schaltungen, H. Bernstein
Auf über 130 Seiten Anwendungsbeispiele mit integrierten Schaltkreisen, Zeitgeber 555, Funktionsgenerator ICL 8038, Opto Elektronik, Operationsverstärker, Festwertspeicher (ROM), u. v. a.
Best.-Nr. 12 **19,80 DM**

HEH, Hobby Elektronik Handbuch **C. Lorenz**
Das Schaltungsbuch f. jeden Hobbyelektroniker, Schaltbeispiele und Bauanleitungen aus dem gesamten Hobbybereich. Lichtorgeln, Alarmanlagen, Eiswarngerät fürs Auto, PLL-Schaltungen u. v. a.
Best.-Nr. 13 **9,80 DM**

Viel **mehr als 33 Programme für den Sinclair SPECTRUM,** R. G. Hülsmann
Ein echt aufregendes Buch für jeden SPECTRUM-Besitzer. Viele Tricks und Tips. 33 Superprogramme wie 3D-Graphik, Crazy Kong, Musik-Computer. Unterprogramme in Maschinensprache, zehn kurze Progr. für den Spectrum mit 16K RAM wie Mondlandung, Spielautomat, Irrgarten, Todeshöhle usw. Sie werden begeistert sein!

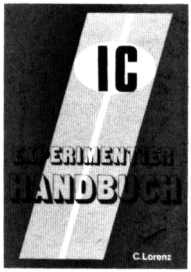
Best.-Nr. 144 **29,80 DM**

Opto-Handbuch, **C. Lorenz**
Das Handbuch für die gesamte Optoelektronik. Eine Einführung und ein ideales Nachschlagwerk. Grundlagen, Definitionen aller Kenngrößen, Opto-Lexikon, Berechnungsgrundlagen, Lichtsender, Lichtempfänger, Anzeigen, Infrarot Detektoren, Optokoppler, Opto-Vergleichsliste, u. v. a.106 Seiten.
Best.-Nr. 15 **19,80 DM**

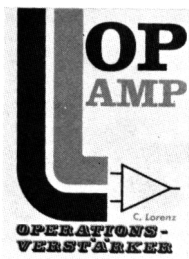
C MOS Einführung, Entwurf, Schaltbeispiele, Teil 1 **H. Bernstein**
Vom C MOS Gatterbaustein über Schieberegister und Zähler bis hin zum C MOS Schreib- Lesespeicher. Insgesamt werden neunzehn interessante und bekannte C MOS Schaltkreise beschrieben. Zu jedem Bauelement sind genaue Daten, Schaltbild und Anwendungsbeispiele angegeben. Im großen Applikationsteil finden Sie: C MOS-Kippstufen, Addierwerke u. Rechenschaltungen, Digital Analog Wandler, Schieberegister für analoge Spannungen, Multiplexsysteme f. analoge Signale u. v. a. Eine komplette Einführung u. gut geeignet für das Selbststudium der C MOS Technik. 140 Seiten.
Best.-Nr. 16 **19,80 DM**

C MOS Entwurf u. Schaltbeispiele, Teil 2, **H. Bernstein**
Fortsetzung von Best-Nr. 16. Anwendungsbeispiele mit genauen Schaltungsbeschreibungen und Bauelementunterlagen. Daten, Anschlußbelegungen weiterer wichtiger hochintegrierter C MOS Elemente. Ein komplettes Arbeits- u. Experimentierbuch. C-MOS Uhrenschaltungen, Schieberegisterschaltungen, Parallel-Serien Umsetzung, statische u. dynamische Speicherschaltungen, Zählschaltungen, Digital Analog-Wandler, Analog Digital Wandler. Digital Voltmeter, I/O Registerschaltungen. RAM und ROM Anwendungen. Über 140 Seiten.
Best.-Nr. 17 **19,80 DM**

C MOS Entwurf u. Schaltbeispiele, Teil 3, **H. Bernstein**
Fortstetzung von Best.-Nr. 17. Eine sehr umfangreiche Applikationssammlung mit hochintegrierten C MOS Elementen. Speicher- und Steuerschaltungen, Multiplex- und Datenbussysteme, Liquid Cristal Anzeigen, Uhrenschaltungen, PLL-Schaltungen, Optoelektronik in Verbindung mit C MOS. Aufbau und Wirkungsweise der Prozeßrechentechnik, Arithmetische Logische Einheiten (ALU) u. andere wichtige Funktionen aus der Prozeßrechentechnik. RAMs, ROMs, und FIFO-Speicherschaltungen.
Best.-Nr. 18 **19,80 DM**

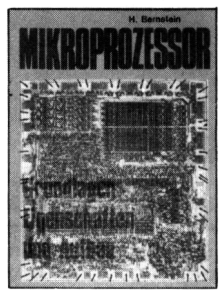


IC Experimentier Handbuch -IC, -EX, C. Lorenz
Eine sehr umfangreiche Schaltungs- und Bauanleitungssammlung mit neuen, jedoch meist beim Fachhandel erhältliche Standard ICs. Rechnerschaltungen, Mikroprozessoren, I/O-Schaltungen, Stoppuhren, druckende und anzeigende Rechner, Digitalvoltmeter, Hilfschaltungen für den Elektronik Experimentierer, A/D-Wandler, Frequenzzähler u. v. a. Viele Schaltungen können auf der IC KIT Experimentierplatine WH-1g aufgebaut werden.
Best.-Nr. 19 **19,80 DM**



Operationsverstärker, C. Lorenz
Dieses Buch umfaßt das gesamte Gebiet der linearen Schaltungstechnik und stellt ein in dieser Preislage bisher noch nie dagewesenes Nachschlagwerk und Einführungshandbuch dar. Bestens geeignet für das Selbststudium. Nach einer pädagogisch geschickt gemachten Einführung folgen theoretische Arbeitsunterlagen und die zugehörigen Schaltbeispiele mit Daten und Gehäuseanschlüssen. Dieses wertvolle Buch dürfte seinen Platz auch bei Ihren Arbeitsunterlagen finden, und wird dann immer von Nutzen sein, wenn es um die Lösung von nicht routinemäßigen Aufgaben geht. Über 150 Seiten.
Best.-Nr. 20 **19,80 DM**

Digitaltechnik Grundkurs, C. Lorenz
Ein Einführungskurs in die Digitaltechnik für Anfänger und Fortgeschrittene. Ein Fachbuch für den programmierten Selbstunterricht. Der ideale Kurzlehrgang für das Selbststudium. Der Kurs vermittelt Ihnen alle wichtigen Grundkenntnisse vom TTL-Gatter bis zum Mikroprozessor und Lösung von Schaltungsaufgaben durch Software. Viele Versuchsaufbauten u. Experimente aus diesem Kurs können auf der IC-KIT Platine WH–1g durchgeführt werden. Grundlagen, Gatter, Zähler, programmierbare Zähler, IC-Tester, Schieberegister, Speicher, Mikroprozessoren u. v. a.
Best.-Nr. 21 **19,80 DM**



Mikroprozessoren, Eigenschaften u. Aufbau, Teil 1, H. Bernstein
Grundlagen, Eigenschaften u. Aufbau von Mikroprozessoren. Organisation von Recheneinheiten und Mikroprogr. Programmierung und Klassifizierung v. Mikroprozessoren. Ablaufdiagramm, Flußdiagramm. Ein Cip-Technik und Multi Chip-Technik, Transfer- und Sprungfunktionen. Speichertechnik: RAMs ROMs, FIFO, FILO. Programmierbare logische Arrays (PLA). Anwendungsbeispiele u. Anwendungsbereiche. Über 120 Seiten.
Best.-Nr. 22 **19,80 DM**

Elektronik Grundkurs, C. Lorenz
Eine leichtverständliche und pädagogisch geschickt gemachte Einführung in die Technik der elektronischen Schaltungen. Ein Kurzlehrgang und Schnellkurs zugleich. Aber auch ein recht brauchbares Nachschlagwerk für den fortgeschrittenen Elektroniker. Mit wenig Mühe können Sie sich hier die Grundkenntnisse der elektronischen Schaltungspraxis aneignen. Das Buch schafft die Voraussetzungen für ein erfolgreiches und sicheres Arbeiten mit interessanten Schaltkreisen modernster Technologien. Unentbehrlich f. das Experimentieren mit den heutigen modernen hochintegrierten Schaltkreisen.
Best.-Nr. 23 **9,80 DM**

ohne Abbildung

Programmieren in Maschinensprache mit Z-80 – Band II, Dr. Schmitter
Dieses sehr interessante Werk geht insbesonders auf TRS-80 und Video Genie ein. Es kann jedoch grundsätzlich auf für alle anderen Personalcomputer, die auf der Z80 CPU basieren, verwendet werden (Color Genie, ZX-81, Spectrum, Osborne, Sharp, usw.). Aus dem Inhalt: µC, µP, RAM, ROM - eine Einführung, Unsere ersten Programmierschritte, Schleifen, Flaps und Sprünge, PUSH, POP, Stapel und Unterprogramme, Input-/Output-Programmierung und Cassettenport. Arithmetik, relative Sprünge und logische Verknüpfungen. BASIC und Maschinencode, wichtige Hilfsmittel wie Disassembler und Assembler.
Best.-Nr. 24 **29,80 DM**

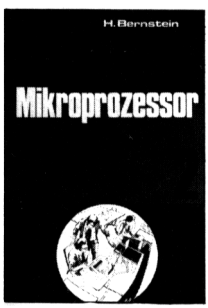
Achtung:
Unter Best.-Nr. 24 erschien bis vor kurzem das Buch Microcomputer-Technik von Blomeyer. Dieses ist vergriffen und wird nicht mehr neu aufgelegt.

ohne Abbildung

68000 Microcomputer Einführung
Eine leicht verständliche Anleitung zur Programmierung des leistungsfähigen 16 Bit Microcomputers von Motorola.
Erscheint ca. Mitte 1983.
Best.-Nr. 25 **39,00 DM**

Achtung:

Lieber Leser,
die Bestell-Nr. 25 und 27 waren früher Hobby Computer Handbuch (25) und Software Handbuch (27). Diese Titel sind vergriffen und werden vorerst nicht mehr neu aufgelegt.



Mikroprozessor, Teil 2, H. Bernstein
Die Fortsetzung von Best.-Nr. 22. Technologie von Mikroprozessor- und Speicherbausteinen. Festwertspeicher, PROM, EPROM, FIFO, Schieberegister, MPR-, ARL- und SAR-Register. Aufbau eines Mikroprozessorsystems mit 8080, RAM- und ROM-Schnittstellen. Befehlssatz 8080. Über 120 Seiten.
Best.-Nr. 26 **19,80 DM**

BASIC-M für Motorola EXORset®
Anwender Handbuch
Eine ausgezeichnete und professionelle Einführung in die BASIC-Programmiersprache. Ideal für die Industrie, aber auch ein ausgezeichnetes Werk für den Anfänger. Diese BASIC-Einführung mit vielen Beispielen gehört zu den besten deutschsprachigen Werken auf diesem Gebiet. Der Stoff geht auf den BASIC-M - Dialekt ein, ist aber auch auf jeder andere BASIC-Version anwendbar.
Best.-Nr. 27 **29,80 DM**

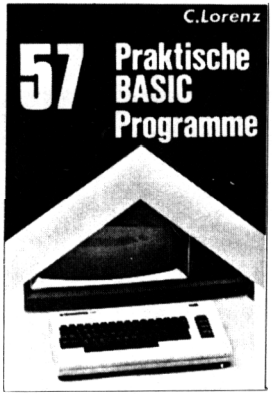
Microcomputer Lexikon u. Wörterbuch von A – Z, C. Lorenz
Englisch/Deutsch – Der Fachausdruck wird übersetzt, ausführlich erklärt und erläutert. Deutsch/Englisch – Übersetzung des Fachausdrucks. Ein Hilfs- und Arbeitsbuch für jeden, der sich heute mit der modernsten Elektronik beschäftigt. Viele engl. Ausdrücke werden heute in der Elektronik, Computer- und Mikroprozessortechnik verwendet und oft fehlt uns eine genaueste und präzise Erläuterung. Ein Lexikon und Wörterbuch in einem einzigen Buch vereint.
Best.-Nr. 28 **29,80 DM**

Microcomputer Datenbuch - (englisch) – ELCOMP
Endlich es es da ! Viele unserer Leser haben seit langem auf dieses Buch gewartet. – Und es hat sich gelohnt. Auf über 800 Seiten haben wir hier für Sie die wichtigsten (fast alle) Bauelemente zusammengestellt (Daten, Anschlußbilder etc.), die sich in den heutigen Personalcomputern befinden. Wir haben alle wichtigen Systeme durchforstet und die Bauelemente herausgesucht. TTL, CMOS, Linear, Spannungsregler, CPU-Schaltkreise, 6502, Z80, 8080, 8085, 8086, 1802, 2650, Z8 u. v. a.
Best.-Nr. 29 **49,– DM**

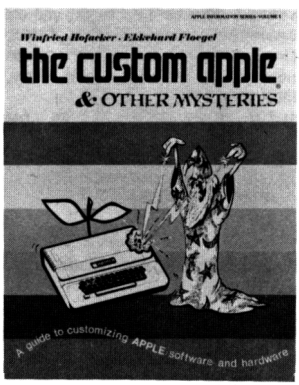
ohne Abbildung

Floppy Disk Selbstbau-Handbuch
von E. Flögel
Dieses Buch soll Ihnen auf einfachste Weise erklären, wie Sie Ihren Personalcomputer mit möglichst geringen Kosten mit einer Floppy Disk Station ersetzten können. Genaue Bauanleitung mit Software, Hinweise und Tips. Multiuser und Mutlitasking macht die Sache ganz besonders interessant.
Erscheint ca. Ende 1983
Best.-Nr. 30 **49,00 DM**

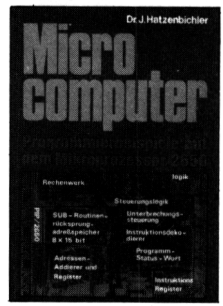
Achtung:
Unter Best.-Nr. 30 wurde früher ein Buch "Aktivtraining" angeboten. Dieses ist vergriffen und wird nicht mehr neu aufgelegt.



57 Programme in BASIC, Lorenz
Ein Buch mit techn.-wissenschaftlichen Programmen u. einer großen Anzahl von Spielprogrammen in BASIC (Games). Ein Buch für jeden, der sich mit dem faszinierenden Hobby der Mikrocomputertechnik befassen will. Alle Listings sind in BASIC und können auf den meisten Personal Computer Systemen gefahren werden.
Best.-Nr. 31 **39,00 DM**

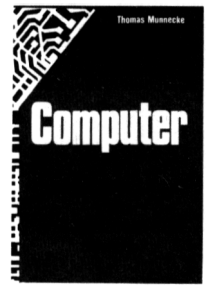


The Custom Apple & other Mysteries, von W. Hofacker und E. Flögel
Dieses Buch bringt auf 190 Seiten Großformat eine Vielzahl von Erweiterungsprojekten für Ihren APPLE II. 6522 Ein-/Ausgabekarte, Tonerzeugung mit AY-3-8912, Analog/Digital-Wandler, EPROM-Burner, Anschluß des 8253 von Intel an den APPLE II, Schrittmotorsteuerungen mit APPLE II in BASIC, PASCAL und FORTH. Zu den einzelnen Projekten sind die notwendigen Platinenvorlagen sowie die Software im Buch enthalten. Alle Platinen zum Buch können vom Hofacker Verlag sofort ab Lager bezogen werden.
Best.-Nr. 680 (englisch) **79,00 DM**



Mikrocomputer Programmierbeispiele für 2650, Dr. J. Hatzenbichler
Eine Einführung in die Programmierung von Mikrocomputern anhand des Prozessors 2650 von Signetics. Viele Programmierbeispiele in Maschinensprache, die Sie auf einem preiswerten Mikroprozessorsystem MIKIT 2650-P2 ausführen können. Zeitschleifenprogr., Blinkschaltung, Lauflicht, Stufenzähler, Stopuhr, Reaktionszeittester, u. a. Zu diesem Buch ist auch ein komplett aufgebautes und getestetes Mikrocomputersystem erhältlich, auf dem Sie alle beschriebenen Programme selbst ausführen können. Über 120 Seiten. (Nur solange Vorrat reicht, wird nicht mehr nachgedruckt.)
Best.-Nr. 33 **19,80 DM**

TINY BASIC Handbuch, Hermann
Das erste deutschsprachige Handbuch über Tom Pittman's TINY BASIC. Eine Einführung in die TINY BASIC-Programmiersprache. Wie kann ich meinen Computer (KIM-1) erweitern und BASIC programmieren. Systemvorschläge. Viele Programmierbeispiele, Tricks und Kniffe.
Best.-Nr. 34 **19,80 DM**



Der freundliche Computer – Was können Sie mit einem Personal Computer anfangen? T. Munnecke
Das Buch soll Ihnen auf die im Titel gestellte Frage eine ausführliche Antwort geben. Es eignet sich für alle, die bisher viel über Mikros gehört haben und gerne ausführlicher Bescheid wissen möchten. Viele interessante Fakten – Welche Computersprache ? Welche Anwendungen ? Welches Gerät soll ich mir kaufen ? 153 Seiten.
Best.-Nr. 35 **29,80 DM**

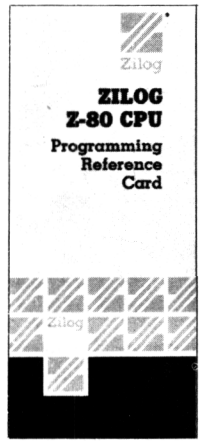
ohne Abbildung

Microcomputer und Roboter
Ein Buch für denjenigen, der sich externe Schaltungen für Microcomputer bauen möchte, die roboterartige Funktionen ausführen können. Spracherkennung, Analog-/Digital-Wandler, Digital-/Analog-Wandler, Ultraschallsensoren, Lichtschranken, Tonerzeugung u. v. a. Erscheint Anfang 1982.
Best.-Nr. 36 **29,80 DM**

ohne Abbildung

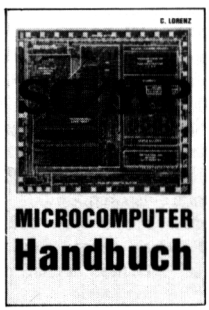
Oszillographen Handbuch
Ein Buch für jeden, der seinen Oszillographen optimal nutzen will. Der Anfänger, der noch nie einen Oszillographen benutzt hat, wird auf einfache Weise mit der Technik und Handhabung vertraut gemacht. Auch die Anwendung in Zusammenhang mit den modernen Mikrocomputersystemen wird beschrieben. Oszillograph als alphanumeres Darstellgerät u. v. mehr. Erscheint ca. Anfang 1982.
Best.-Nr. 103 **19,80 DM**

Z80 Assembler Handbuch
Dieses Handbuch braucht jeder, der ein Z80 Computersystem mit Z80 CPU hat (TRS-80, NEC, ALTOS, Sinclair, Video Genie usw.). Auf über 420 Seiten wird jeder Maschinenbefehl erklärt.
Best.-Nr. 8029 **29,80 DM**



Z-80 CPU Programming Reference Card
Eine praktische und kompakte Zusammenfassung der Z-80-Befehle, Z-80 Pinout, Registeraufbau, ASCII Zeichensatz kompletter Befehlssatz.
Best.-Nr. 252 **5,00 DM**

CMOS-Experimentierbuch
Eine kleine Einführung in die CMOS Schaltungstechnik. CMOS Grundlagen, Behandlungshinweise für CMOS Bausteine, Zusammenst. der wichtigsten CMOS Bausteine und deren Anschlußbilder. Viele praktische CMOS Schaltbeispiele. Ideal für jeden Elektroniker.
Best.-Nr. 106 1 **5,00 DM**



Handbuch für SC/MP
Ein echtes Handbuch für SC/MP (INS8060) Microcomputerbesitzer und solche die es werden wollen. Komplette Einführung mit vielen Schaltbeispielen, Schaltbildern und Programmlistings in Maschinensprache und TINY BASIC. Komplettes ELBUG-Listing, CPU-Karte, RAM-Karte, ROM-Karte, ROM-Programmierer, Cassetten-Interface, Fernsehinterface, Netzteil, Hex Ein-/Ausgabe, SC/MP Einkartenmicrocomputer u. v. a. 330 Seiten.
Best.-Nr. 108 **29,80 DM**

6502 Microcomputer Programmierung, **P. Heuer**
Eine deutschsprachige Einführung in die Maschinensprachenprogrammierung anhand des 6502 Microcomputers. Ein echtes Anleitungsbuch zum Einstieg in die Microcomputertechnik mit Hilfe des KIM-1. Viele Programmierbeispiele, die von einem Pädagogen speziell für Anfänger entwickelt wurden. Auch PET, AIM, SYM und ATARI-Besitzer brauchen dieses Buch.
Best.-Nr. 109 **29,80 DM**

Programmierhandbuch für PET
C. Lorenz
Ein für den PET produziertes Buch. Es beginnt da, wo Ihr mitgeliefertes Handbuch aufhört. Einführung Maschinenprogrammierung, Assembler, Ein-/Ausgabeprogrammierung, Programmiertricks, Analog / Digital-Wandler, Graphik, Spracherkennung u. v. a. mehr. Viele Listings, die Sie selbst eintippen können. Viele Informationen eignen sich auch für VC–20-Besitzer. 324 Seiten.
Best.-Nr. 110 **29,80 DM**

Programmieren mit TRS-80, Stübs
Das erste in einem deutschen Verlag produzierte Buch über den erfolgreichen Personal Computer von TANDY. Ein Buch für jeden, der einen TRS-80 bereits besitzt oder vor der Entscheidung steht, welchen Computer er sich anschaffen soll. Einführung, Programmiertricks, Erweiterungen, Maschinenprogrammierung und viele Programme (Listing mit Beschreibung). 202 Seiten.
Best.-Nr. 111 **29,80 DM**

BASIC Programmierhandbuch
Einführung und Nachschlagewerk. Speziell für die BASIC-Versionen der modernen Microcomputersysteme. Jeder Befehl wird ausführlich beschrieben und ein Beispielprogramm gezeigt. Sehr übersichtlich und praktisch. Am Schluß finden Sie ein komplettes BASIC-Programm, das Ihnen über einen Computer BASIC lehrt.
Best.-Nr. 113 **19,80 DM**

Der Microcomputer im Kleinbetrieb
Das Buch für jeden Geschäftsmann. Auf über 170 Seiten erfahren Sie, was Sie als Gewerbetreibender oder freiberuflich Tätige über Microcomputer und die Anwendung wissen sollten. Geschichtlicher Hintergrund Geräteauswahl, Beispiele aus der Praxis, Programmbeispiele wie z. B. Textverarbeitung, Reisebüro, Ladenkasse, Adressverwaltung, u. v. a. Betriebswirtschaftliche Auswertung, Finanzbuchhaltung, Erfolgsanalyse mit dem Microcomputer, Liquiditätsrechnung, kurzfristige Erfolgsrechnung, Microcomputer für Freiberufler, Grundlagen der Finanzbuchhaltung für Microcomputeranwender. Dieses Buch kann Ihnen als Geschäftsmann für die Zukunft tausende einsparen.
Best.-Nr. 114 **39,80 DM**

PASCAL Handbuch, E. Flögel
Von BASIC zu PASCAL. Ein Einführungs- Lehr und Arbeitsbuch für jeden der sich mit PASCAL beschäftigen will oder muß. Viele Programmbeispiele, viele Tricks wie PEEK und POKE, Einbinden von Maschinenprogrammen u. v. a.
Best.-Nr. 112 **29,80 DM**

16 Bit Microcomputer, J. Koller
Einführung, Daten, Eigenschaften, Anwendungen. Dieses Werk ist eine echte Sensation !
Alle 16 Bit Prozessoren werden beschrieben und erläutert. Applikationsbeispiele, Programmierhinweise. TMS 9900, 8086, Z8000, MC 68000, NS 16000, IAPX 486, IAPX 432. Über 370 Seiten.
Best.-Nr. 116 **29,80 DM**

ohne Abbildung

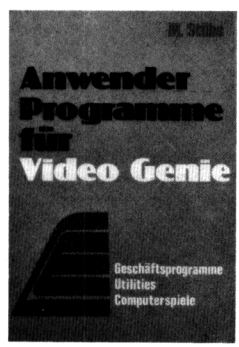
FORTRAN für Heimcomputer
Einführung in die FORTRAN-Programmiersprache mit vielen Beispielen. Grundsätzliches über die verschiedenen Microcomputersysteme, die bereits mit FORTRAN-Compiler lieferbar sind. Allgemeine Übersicht, Tips und Hinweise.
Erscheint ca. Ende 1982 .
Best.-Nr. 117 **19,80 DM**

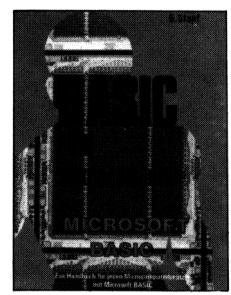
Programmieren in Maschinensprache mit 6502, E. Flögel, W. Hofacker
Das deutschsprachige Werk über 6502 Maschinenprogrammierung. Einführung, Grundlagen, Eigenschaften, Adressierungsarten, Befehlsarten. Wie entwickelt man ein 6502 Maschinenprogramm? Handassemblierung, viele Programmbeispiele mit genauen Angaben direkt zum Eingeben in den Apple II mit Adressangabe (keine blutleeren Beispiele ohne Adresse), Verwendung von Assemblern. AIM-Assembler, Disassembler, Relocator, 6522 VIA, 6520, Interrupt, Fehlersuche in Maschinenprogrammen, Maschinensprache, Programmiertricks. Spezielle Abschnitte für Maschinensprachenprogrammierung über PET, CBM 3000, CBM 4000 u. VC-20, ATARI 400/800, Apple II, AIM sowie Ohio Scientific Challenger. Dieses Buch sollte jeder 6502 Systemanwender besitzen. Ca. 240 Seiten.
Best.-Nr. 118 **49,00 DM**

BASIC für Fortgeschrittene
Endlich ein BASIC-Buch für den fortgeschrittenen Programmierer. Alle wichtigen Befehle aus der Stringmanipulation, Disk-Befehle, WAIT, INSTR, WHILE WHEND usw. werden an Beispielen besprochen. Alle Befehle sind übersichtlich wie in einem Nachschlagwerk mit großen Überschriften angeordnet. Dann folgt ein umfangreicher BASIC Kurs für Fortgeschrittene mit vielen Beispielen (Inventur, Rechnungen schreiben, Adressenverwaltung usw.). Am Schluß finden Sie dann noch einen Vergleich der wichtigsten Sortiermethoden sowie ein Programm zur Vorhersage von Ereignissen.
Best.-Nr. 122 **39,00 DM**

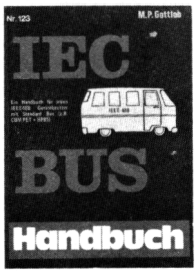
Programmieren in Maschinensprache (Z80), C. Lorenz
Eine sehr ausführliche Einführung in die Z80 Maschinensprache mit vielen Beispielen. Die Beispiele können mit Hilfe des TRS-80 Level II sowie dem T-BUG von TANDY und den T-BUG-Erweiterungen (IN LOCO, T-STEP, T-LEGS) ausgeführt werden. Ein unentbehrliches Buch für jeden, dem die BASIC-Programmiersprache von der Geschwindigkeit her zur Lösung seiner Aufgaben nicht mehr ausreicht.
Best.-Nr. 119 **39,00 DM**

Anwenderprogramme für TRS-80 von Martin Stübs
Ein Buch, voll mit interessanten Anwenderprogrammen für TRS-80 Level II 16K und Video Genie (teilweise Diskette u./od. Cassette). Hauptsächlich Programme für den Manager, Geschäftsmann, Klein- und Mittelbetrieb. Auch einige interessante Spiele sind enthalten. Terminkalender, Reservierungsprogramm für Omnibusunternehmen und Hotels, Textverarbeitung, usw.
Best.-Nr. 120 **29,80 DM**





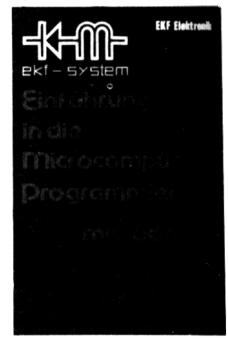
Microsoft BASIC-Handbuch
Die deutsche Übersetzung des erfolgreichen Microsoft BASIC-Handbooks. Leicht verständliche Einführung mit vielen interessanten Programmbeispielen. Das kompetente Werk von Microsoft selbst. Ideal als Zusatzliteratur zu jedem BASIC-Buch.
Best.-Nr. 121 **29,80 DM**



IEC Bus-Handbuch, M. P. Gottlob
Ein Handbuch und Nachschlagewerk für alle Besitzer von Computern mit IEC (IEEE 488 Bus). Dazu gehört auch der PET sowie alle CBM-Computer. Grundlagen, das BUS-System, Meßdatenübertragung, Adressierung eines Instruments, kl. IEC-BUS-Lexikon u.v.a.
Best.-Nr. 123 **19,80 DM**

Programmieren in Maschinensprache mit CBM
An Hand eines praktischen Beispiels (Sortierroutine) wird der Unterschied zwischen BASIC und Maschinenprogrammen gezeigt. Das Maschinenprogramm kann mit dem leistungsfähigen MONJANA/1 Monitor in ROM erstellt werden. Am Schluß finden Sie weitere wichtige Informationen wie Dez/Hex-Umrechnungstabelle, Befehlslisten, ASCII-Tabelle sowie eine ROM-Vergleichsliste zwischen 8k PET und den neuen CBM-Maschinen.
Best.-Nr. 124 **19,80 DM**

MONJANA Monitor im ROM
Best.-Nr. 1241 **79,00 DM**



Einführung in die Microcomputer Programmierung mit 6800
Eine sehr gute Einführung in die Microcomputertechnik mit Hilfe des Mikroprozessors 6800. Ausführliche Erklärungen mit vielen Beispielen und Anleitungen. Theoretische Grundlagen. CPU-Architektur, Befehlssatz, Systemaufbau, Hilfsmittel der Programmierung, Trainingsprogramme, Systemkomponenten, FIRMWARE. Ein komplettes Monitorprogramm (Betriebssystem) ist als Listing enthalten. Über 250 Seiten.
Best.-Nr. 127 **49,00 DM**

Programmierbeispiele für CBM
Ein Buch mit vielen BASIC-Programmen für CBM und PET. Spiele, Geschäftsbereich, Erziehung und Wissenschaft, Utilities, Hilfen für Maschinensprachenprogrammierung, trickreiche Programme. Viele Programme für wenig Geld.
Best.-Nr. 130 **19,80 DM**

ohne Abbildung

ELCOMP
Fachzeitschrift
für Microcomputertechnik

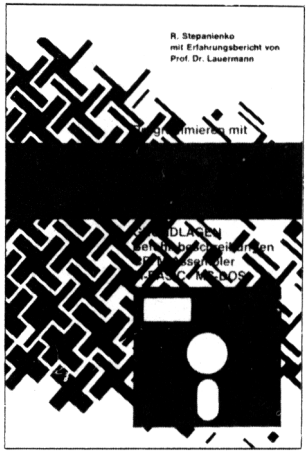
Die kompetente Fachzeitschrift für das moderne Gebiet der Microcomputertechnik. Erscheint 2 x pro Jahr. Preis pro Heft 29,80 DM incl. MwSt., Porto und Verpackung. Wer die neuesten Informationen aus diesem Gebiet für sich nützen möchte, muß ELCOMP lesen. Software, Technische Tips, Programmiertricks, Bauanleitungen, Systembeschreibungen, u. v. a.
Best.-Nr. 125 **29,80 DM**

ohne Abbildung

Programmieren mit dem CBM
Ein Hand- und Programmierbuch für alle CBM-Besitzer der 3000, 4000 sowie der 8000er Serie. Viele Tricks und Programmierbeispiele, Anleitungen.
Best.-Nr. 128 **29,80 DM**

ohne Abbildung

ELCOMP Leser Programmierhandb.
Hier fassen wir die besten Programme unserer ELCOMP-Leser zusammen. Programme für PET, CBM, TRS-80, AIM, Superboard, C4P, Exidy, Sharp, MZ80K, Apple II, Nascom I und II und TI 99/4 werden als Listing mit kurzer Beschreibung allen Lesern zugänglich gemacht. Erscheint Anfang 1982.
Best.-Nr. 129 **69,00 DM**



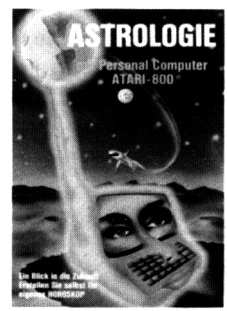
CP/M Handbuch
Grundlagen, Einführung, Hilfs- und Handbuch für jeden der mit dem "Software-Bus" arbeiten möchte. Ideal auch für Anfänger. Praktisches Handbuch für den Profi. Erscheint ca. Ende 1981.
Best.-Nr. 132 **19,80 DM**

FORTH Handbuch und Einführung von E. Flögel, W. Hofacker
FORTH ist nicht nur eine sehr leistungsfähige Programmiersprache – es ist schon fast eine "Religion". FORTH eignet sich bestens für industrielle Steuerungen, Grafik etc. Grundlagen und viele Programmbeispiele (Apple II, Ohio, ATARI usw.). Erscheint Mitte 1982.
Best.-Nr. 137 **39,00 DM**

BASIC für blutige Laien, Matthaei
Endlich ein Buch für den Anfänger und Laien. Ziel dies Buches ist es, dem "blutigen Laien" die Grundlage der Programmiersprache BASIC zu vermitteln. Lernen wird nun zum echten Vergnügen und Freizeitspaß. Auch der Preis macht Spaß.
Best.-Nr. 139 **nur 19,80 DM**

Programmieren in BASIC und Maschinencode mit dem ZX81, E.Flögel
Ein Buch für Sinclair ZX81 Besitzer und solche die es werden wollen. Was heißt Programmieren?, Programmerstellung in BASIC, Spiele, Spielelemente, Programme für die Schule, Datenverwaltungsprogramme, Lagerbestand, Schallplattenverzeichnis, Programmieren in Z80 Maschinensprache, Anschluß einer PIO und externer Schaltungen, Lösen von digitalen Steuerungsaufgaben mit dem ZX81 u. v. a. Dieses Buch gehört auf den Tisch eines jeden ZX81 Besitzers (mehr als 20 komplette Programme, Maschinensprachen-Monitor, etc.).
Best.-Nr. 140 **29,80 DM**

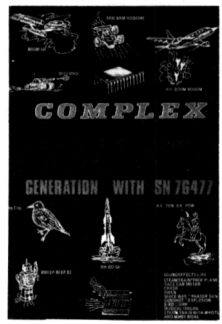
Programme für den VC-20 von W. Hofacker
Ein Buch des Hofacker Verlages speziell für den VC-20 Volkscomputer von Commodore. Viele komplette Programme wie Wortprozessor, Maschinensprachenmonitor, lustige Spiele, Programmieren in Maschinensprache, Ein-/Ausgabeprogrammierung. Wichtige Adressen des Betriebssystems, Tabellen, Speichererweiterungen, Dual-Joystick Bauanleitung ú. v. a. Sie werden begeistert sein.
Best.-Nr. 141 **29,80 DM**



Astrologie mit dem Personal-Computer ATARI 800, W. Hofacker
Ein Blick in die Zukunft. Erstellen Sie selbst Ihr eigenes Horroskop. Dieses Büchlein zeigt dem interessanten Leser wie man sein eigenes Horroskop mit professioneller Genauigkeit berechnen kann. Was braucht man dazu? Wie geht man vor? Was hat es mit der Astrologie auf sich? Was braucht man für die Deutung? Enthält ein komplettes Listing in BASIC und Ma.-Code (ATARI 800, 48K RAM + Disk).
Best.-Nr. 175 **49,00 DM**

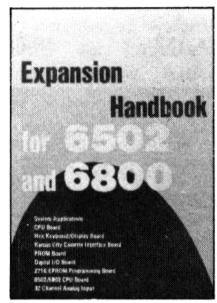
ELCOMP-Bücher Englisch

Care and Feeding of the Commodore PET
Das ideale Buch für den Hardware-Bastler. Viele Tricks, Schaltbilder, Hinweise und Erläuterungen für den, der gerne selbst Erweiterungen bauen möchte. Memory Map für 8k PET und CBM, Bauanleitung für eine serielle Schnittstelle u. v. a.
Best.-Nr. 150 **19,80 DM**

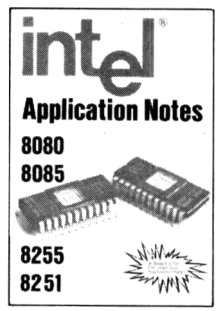


Complex Sound Generation with SN 76477
Ein Applikationsheft für einen der interessantesten integrierten Bausteine unserer Zeit. Ein LSI-Baustein zur Tonerzeugung. Je nach äußerer Beschaltung können Sie mit diesem Baustein die verrücktesten Töne erzeugen. Dampfeisenbahngeräusch mit Dampfpfeife, Vogelgezwitscher, Hundegebell, elektronische Orgel, Schuß mit Explosion u. v. a. mehr.
Best.-Nr. 154 **9,80 DM**

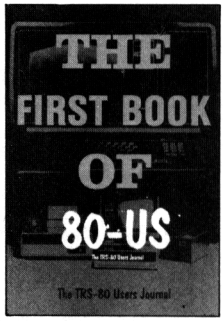
Expansion Handbook for 6502 and 6800
Das ideale Handbuch für alle KIM, SYM, AIM, PET und Challenger Computer-Freunde. Das Buch beschäftigt sich ausschließlich mit dem S-44-Bus. Dies ist exakt der Bus von SYM, AIM und KIM. Sehr viele Schaltbilder: CPU-Platine, Hex-Tastatur Eingabe, Knsas City Interface, RAM u. ROM-Karte, Analog-Eingabe Board u. v. a. Das Buch ist für jeden 6502 Systembesitzer unentbehrlich. Ca. 150 Seiten.
Best.-Nr. 152 **19,80 DM**



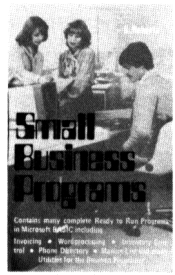
Microsoft 8k BASIC Reference Manual
Eine sehr gute BASIC-Einführung. Auch als Handbuch zum Nachschlagen bestens geeignet. Ideal für jeden PET, CBM, TRS-80, KIM-BASIC, SYM-BASIC, AIM- und APPLE-Besitzer. 73 Seiten DIN A4 mit vielen Beispielen.
Best.-Nr. 151 **9,80 DM**



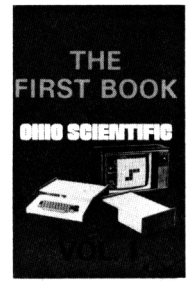
Intel Application Notes (8080, 8085, 8255, 8251)
Dieses Buch braucht jeder, der mit 8080, 8085 oder Z-80 Mikroprozessoren arbeitet.
Wir haben die interessantesten Applikationsberichte in diesem Buch zusammgefasst. Aus dem Inhalt: Designing with Intel's Static RAM's 2102, Memory Design with the Intel 2107B. 8255 Programmable Peripheral Interface Applications, Using the 8202 Dynamic RAM Controller u. v. a.
Best.-Nr. 153 **29,80 DM**



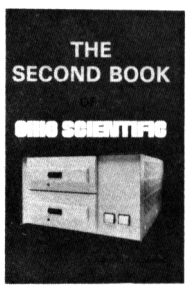
The First Book of 80-US
Für den TRS-80 Freund eine echte Preissensation. Die ersten fünf Hefte aus 80-US Journals in einem Sammelband zusammengefaßt. Voll mit vielen sehr interessanten Hard- und Softwareideen, Tricks. Viele komplette Programmbeispiele (Listings) in BASIC u. Z-80 Maschinensprache. Über 250 Seiten DIN A4. Farbiger Umschlag. Dieses Buch sollte jeder TRS-80 Besitzer oder der es werden will im Schrank haben.
Best.-Nr. 155 **29,80 DM**
(solange Vorrat reicht)



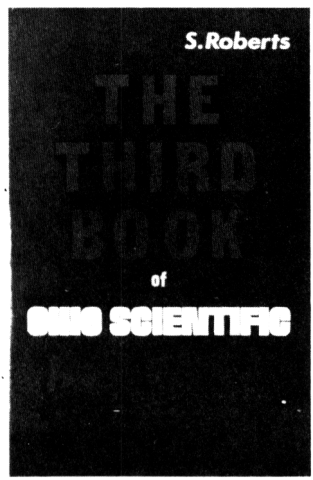
Small Business Programs, S. Roberts
Ein Buch für denjenigen, der die modernen Microcomputer (speziell TRS-80, Apple, PET, North Star, Challenger) zur Rationalisierung in seinem Klein- oder Mittelbetrieb einsetzen möchte. Viele nützliche Tips, Hinweise und Programmierbeispiele. Dieses Buch sollte jeder Geschäftsmann u. Microcomputerfreund besitzen.
Best.-Nr. 156 **29,80 DM**



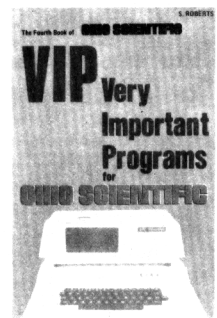
The first Book of Ohio Scientific
Das erste weltweit produzierte Buch für die erfolgreiche Ohio Scientific Challenger Computerserie. Grundlagen, viele Programmiertricks Hardwaretips, Umbauanleitungen, Programmierbeispiele u. v. a. Glanzumschlag, 186 Seiten.
Best.-Nr. 157 **19,80 DM**
(solange Vorrat reicht)



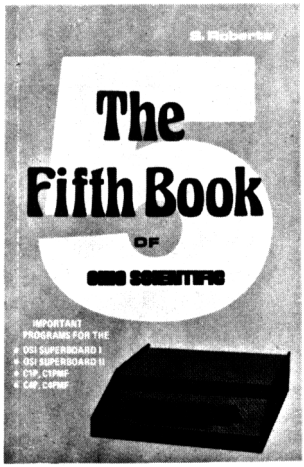
The second Book of Ohio Scientific
Eingehende Beschreibungen über praktische und geschäftsorientierte Software. Speicher Test Programm, Tricks und Tips für Disketten-Anwender. Mini-Floppy-Expansion u. v. a. 159 Seiten.
Best.-Nr. 158 **19,80 DM**



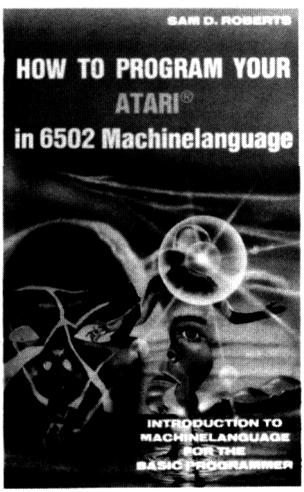
The third book of Ohio
Wie erweitere ich mein Challenger System ? Universelle I/O-Karte, EPROM-Burner für 2716, EPROM, RAM-Karte, 6522 VIA-Karte. Wo notwendig mit kompletter Software. Dieses Buch braucht jeder Ohio-Benutzer. Komplette Schaltbilder und Aufbauhinweise.
Best.-Nr. 159 **19,80 DM**



The fourth Book of Ohio Scientific
Ein Buch voll mit Programmen für das Superboard, C4P, C4PMF und C28P. Die Softwarequelle für jeden Challenger-Fan. Alle Programme sind getestet und auch auf Cassette verfügbar. 170 Seiten Listings und Beschreibungen.
Best.-Nr. 160 **29,80 DM**
Best.-Nr. 8324 Cassette 29,80 DM

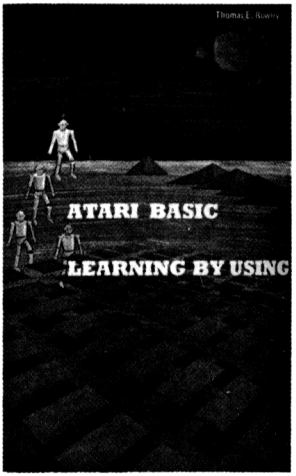


The Fifth Book of Ohio Scientific
Kein anderer Verlag der Welt hat fünf Bände für den so erfolgreichen und leistungsfähigen Personalcomputer bisher produziert. Der 5 Band bringt wieder eine Vielzahl von sehr interessanten BASIC- und 6502 Maschinenprogrammen. Neben einer Adressenverwaltung, einem Textprogramm, Fakturierprogrammen und Utilities finden Sie auch wieder viele interessante Spiele sowie eine komplette Abhandlung über Start- und Landesimulationen, die sich besonders als Hilfsmittel zur Entwicklung eigener Mondlandespiele eignen. Dem Buch kann vom erfahrenen BASIC-Programmierer auch viel Stoff zur Implementation f. andere Rechner entnommen werden (englisch).
Best.-Nr. 161 **19,80 DM**



How to program your ATARI in 6502 Machinelanguage, Roberts
Eine sehr gute Einführung in die 6502 Maschinensprache mit dem ATARI 400/800.
Im einzelnen leicht verständlich und überschaubaren Lektionen werden Sie an Hand von praktischen Beispielen in die wunderbare Welt der Maschinensprache geführt. Sehr viele interessante Programmbeispiele für den ATARI. Tricks, Kniffe und Tips.
Best.-Nr. 169 **29,80 DM**

ZX-81 / TIMEX
Programming in BASIC and Machine Language by E. Flögel
Ein Buch mit vielen Tips und Programmen für den erfolgreichen Sinclair ZX-81 Personalcomputer. Dieses Buch ist eine Übersetzung unseres Titels Nr. 140.
Sehr interessant ist die Anleitung zum Aufbau einer Z-80 PIO an den ZX-81.
Best.-Nr. 174 **29,80 DM**



ATARI BASIC-Learning by USING v. Thomas E. Rowley (engl.)
Ein echtes Action-Buch für Ihren ATARI 400/800. Hier findet mehr statt als nur lesen. Sie verwenden es und machen neue Entdeckungen. Viele nützliche Routinen und Hilfsprogramme. Grafik, Tonerzeugung, Joystickprogrammierung, PEEK und POKE und Special-"Struff".
Best.-Nr. 164 **19,80 DM**

Programming in 6502 Machinelanguage for PET + CBM
Dieses Buch enthält eine große Auswahl sehr wertvoller Informationen für den PET und CBM Besitzer der 3000er und 2000er Serie. Neben dem Listing und Beschreibung für einen sehr leistungsfähigen EDITOR/Assembler in Maschinensprache mit Beispiel für den Sie einen Disassembler, Linker, Editor, Assembler in BASIC und Maschinencode sowie einen kompletten Maschinensprachenmonitor für PET 2001. (engl.)
Best.-Nr. 166 **49,00 DM**

Hiermit gebe ich folgende Bestellung unter Anerkennung Ihrer Lieferungs- und Zahlungsbedingungen auf.

ELCOMP

Menge	Artikelbezeichnung	Einzelpreis

Menge	Artikelbezeichnung	Einzelpreis

Unterschrift (für Jugendliche unter 18 J. der Erziehungsberechtigte) Datum

---

Hiermit bestelle ich die ELCOMP-Sonderhefte bis auf Widerruf.

○ Ich bitte um Abbuchung von DM 29,80 (incl. Versand und Verpackung) von meinem Konto (Giro oder Postscheckkonto)

Kto.-Nr. ..... BLZ .....

Geldinstitut: ..... Ort: .....

○ Ich bitte um Lieferung per Nachnahme. Mit der Lieferung eines Heftes wird der Betrag von DM 29,80 + DM 6,50 NN-Gebühr erhoben.

Name/Vorname

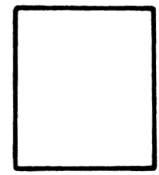
Straße/Haus-Nr.

PLZ Ort

Datum..... Unterschrift .....

ELCOMP

POSTKARTE



Absender
Bitte deutlich ausfüllen

_____
Vorname/Name

_____
Beruf

_____
Straße/Nr.

_____
Plz · Ort

ELCOMP

MIKROCOMPUTER BOOK STORE

Tegernseerstr. 18

D-8150 Holzkirchen /Obb.

ABSENDER:

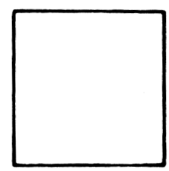
.....
Name, Vorname

.....
Straße

( ) .....
PLZ Ort

.....
Telefon

Die Bestellung der Sonderhefte kann JEDERZEIT widerrufen werden. Die Belieferung wird zum nächst erscheinenden Heft eingestellt.
Eine Kündigung NACH Auslieferung eines Sonderheftes (rückwirkend) ist NICHT möglich.



ELCOMP

Ing. W. Hofacker GmbH
Tegernseer Straße 18
D-8150 Holzkirchen

Leercassetten für Microcomputer

C–10

Die ideale Cassettenlänge für Ihren Personalcomputer.
Praktisch – handlich und betriebssicher

Kassetten mit nur 10 Minuten Spieldauer (2 x 5 Minuten) haben sich zur Aufzeichnung von Daten im Mikrocomputerbereich bestens bewährt.

Vorteile der C-10 Computer Cassette vom HOFACKER Verlag:

- weniger Bandsalat
- kurze Rückspulzeiten
- schnelles Auffinden von Programmen
- bessere Gleichlaufeigenschaften
- einfache Programmverwaltung

Die C-10 HOFACKER Datencassette bietet weiterhin:

- extrem hoch aussteuerbares Bandmaterial (Agfa)
- hochwertiges Cassettengehäuse, 5fach verschraubt
- Tefloneinlage für gute Laufruhe
- Staubdichtes Glasfenster

Die C-10 HOFACKER Datencassette wird seit 1978 speziell für Microcomputeranwender produziert. Die Cassetten bieten ein Höchstmaß an Betriebssicherheit bezüglich fehlerfreier Aufnahme und Wiedergabe.

Hier eine kurze Übersicht über die Anzahl der Bytes, die Sie auf eine C-10 Cassette abspeichern können:

Computer	Speichermöglichkeit	Computer	Speichermöglichkeit
ATARI 400/800	16K	APPLE	36K
Sharp MZ-80	32K	APPLE II	16K
AIM 65	16K	Heathkit	36K
Ohio Scientific	10K	Kansas City Std.	16K
TRS-80	16K	KIM-1	12K
TRS-80 Color Computer	24K	NASCOM	12K
Video Genie	16K	Exidy Sorcerer	12K
Sinclair ZX80/81	16K	SYM-1	12K

BESTELLSCHEIN

Menge	Beschreibung	Preis/DM	Gesamt
	1 Cassette	**3,50**	
	10 Cassetten	**29,80**	
	100 Cassetten	**249,00**	

Lieferanschrift

Name. . . . . .
Straße . . . . . .
Ort . . . . . .

Datum. . . . . Unterschrift . . . . . .

HOFACKER

Ing. W. Hofacker GmbH
Tegernseerstr. 18
D–8150 Holzkirchen
Tel.: (0 80 24) 73 31

Weitere interessante Bücher von Hofacker:

Best.-Nr.	Titel	Preis/DM
	Bücher in deutscher Sprache aus dem Hofacker-Verlag	
1	Transistor Berechnungs- und Bauanleitungsbuch – 1	29,80
2	Transistor Berechnungs- und Bauanleitungsbuch – 2	19,80
3	Elektronik im Auto	9,80
4	IC-Handbuch, TTL, CMOS, Linear	19,80
5	IC-Datenbuch, TTL, CMOS, Linear	9,80
6	IC-Schaltungen, TTL, CMOS, Linear	19,80
7	Elektronik Schaltungen	19,80
8	IC-Bauanleitungs-Handbuch	19,80
9	Feldeffekttransistoren	9,80
10	Elektronik und Radio	19,80
11	IC-NF Verstärker	9,80
12	Beispiele integrierter Schaltungen (BIS)	19,80
13	HEH, Hobby Elektronik Handbuch	9,80
14	IC-Vergleichsliste	29,80
15	Optoelektronik Handbuch	19,80
16	CMOS Teil 1, Einführung, Entwurf, Schaltbeispiele	19,80
17	CMOS Teil 2, Entwurf und Schaltbeispiele	19,80
18	CMOS Teil 3, Entwurf und Schaltbeispiele	19,80
19	IC-Experimentier Handbuch	19,80
20	Operationsverstärker	19,80
21	Digitaltechnik Grundkurs	19,80
22	Mikroprozessoren, Eigenschaften und Aufbau	19,80
23	Elektronik Grundkurs, Kurzlehrgang Elektronik	9,80
24	Progr. in Maschinensprache mit Z80, Band II	29,80
25	68000 Microcomputer Einführung (i. V.)	39,00
26	Mikroprozessor, Teil 2	19,80
27	BASIC-M Anwender-HB f. 6800/09/68000 (Motorola)	29,80
28	Lexikon + Wörterbuch f. Elektr. u. Mikroprozessor	29,80
29	Mikrocomputer Datenbuch	49,80
30	Floppy Disk Selbstbau-Handbuch (i. V.)	49,00
31	57 Programme in BASIC	39,00
33	Microcomputer Programmierbeispiele	19,80
34	TINY-BASIC Handbuch	19,80
35	Der freundliche Computer	29,80
103	Oszillographen-Handbuch	19,80
108	Rund um den Spectrum (Progr., Tips u. Tricks) (i. V.)	29,80
109	6502 Microcomputer Programmierung	29,80
110	Programmierhandbuch für PET	29,80
111	Programmieren mit TRS-80 (Video Genie)	29,80
112	PASCAL-Programmier-Handbuch	29,80
113	BASIC-Programmier-Handbuch	19,80
114	Der Microcomputer im Kleinbetrieb	39,80
115	6809 Programmier Handbuch (i. V.)	49,00
116	Einführung 16-Bit Micorocomputer	29,80
117	FORTRAN für Heimcomputer	19,80
118	Programmieren in Maschinensprache mit dem 6502	49,00
119	Programmieren in Maschinensprache (Z80) Band I	39,00
120	Anwenderprogramme für TRS-80 u. Video Genie	29,80
121	Microsoft BASIC-Handbuch	29,80
122	BASIC für Fortgeschrittene	39,00
123	IEC-Bus Handbuch	19,80
124	Programmieren in Maschinensprache mit CBM	19,80
127	Einführung i. d. Microcmputer-Progr. mit 6800	49,00
128	Programmieren mit dem CBM	29,80
129	ELCOMP-Leser Programmierhandbuch (i. V.)	69,00
130	Programmierbeispiele für CBM	19,80
132	CP/M-Handbuch	19,80
133	Handbuch für MS/DOS (i. V.)	29,80
137	FORTH Handbuch	49,00
139	BASIC für blutige Laien	19,80
140	Progr. i. BASIC u. Maschinencode mit dem ZX81	29,80
141	Progr. für VC-20 (Spiele, Utilities, Erweiterungen)	29,80
143	35 Programme für den ZX81	29,80
175	Astrologie auf dem ATARI 800	49,00
8029	Z-80 Assembler-Handbuch	29,80
	Bücher in englischer Sprache	
	1. Von ELCOMP Publishing, Inc., Los Angeles, CA.	
150	Care and Feeding of the Commodore PET	19,80
151	8K Microsoft BASIC Reference Manual	9,80
152	Expansion Handbook for 6502 and 6800	19,80
154	Complex Sound Generation using the SN76477	9,80
156	Small Business Programs	29,80
158	The Second Book of Ohio Scientific	19,80
159	The Third Book of Ohio Scientific	19,80
160	The Fourth Book of Ohio Scientific	29,80
161	The Fifth Book of Ohio Scientific	19,80
162	ATARI Games in BASIC	19,80
163	The Peripheral Handbook (i. V.)	29,80
164	ATARI-BASIC Learning by Using	19,80
166	Programming in 6502 Machinelanguage PET/CBM	49,00
169	How to Progr. your ATARI in 6502 Machinelanguage	29,80
170	FORTH on the ATARI – Learning by Using	29,80
171	See the Future with your ATARI (Astrology)	49,00
172	Hackerbook I (Tricks + Tips for your ATARI (i. V.)	29,80
173	PD-Program Descriptions (ATARI)	9,80
174	ZX-81/TIMEX Progr. i. BASIC a. Machine Lang.	29,80
176	Programs + Tricks vor VIC's (i. V.)	29,80
177	CP/M – MBASIC and the OSBORNE (i. V.)	29,80
	2. Von IJG Inc., Upland Californien	
240	TRS-80 Disk & other Mysteries	69,00
245	Microsoft BASIC Decoded (TRS-80)	89,00
246	BASIC Faster and Better (TRS-80)	129,00
247	The Custom TRS-80 & other Mysteries	129,00
680	The Custom Apple & other Mysteries	79,00
681	Machine Language Disk I/O (TRS-80)	129,00
	Riesenprogrammsammlung in BASIC	
8048	BASIC Software Vol. VI	199,00
8049	BASIC Software Vol. VII	159,00
8050	BASIC Software Vol. I	99,00
8051	BASIC Software Vol. II	99,00
8052	BASIC Software Vol. III	149,00
8053	BASIC Software Vol. IV	39,00
8054	BASIC Software Vol. V	39,00

Der Hofacker Verlag produziert und vertreibt neben einer sehr großen Auswahl an Fachbüchern für Elektronik und Microcomputertechnik noch:

– Leerplatinen und Bauanleitungen für Zusatzeinrichtungen für Ihren Personalcomputer, sowie
– Programme (Software) für die bedeutenden Personalcomputer.

(i. V. bedeutet: Buch ist in Vorbereitung)

HOFACKER

HOLZKIRCHEN **LOS ANGELES**